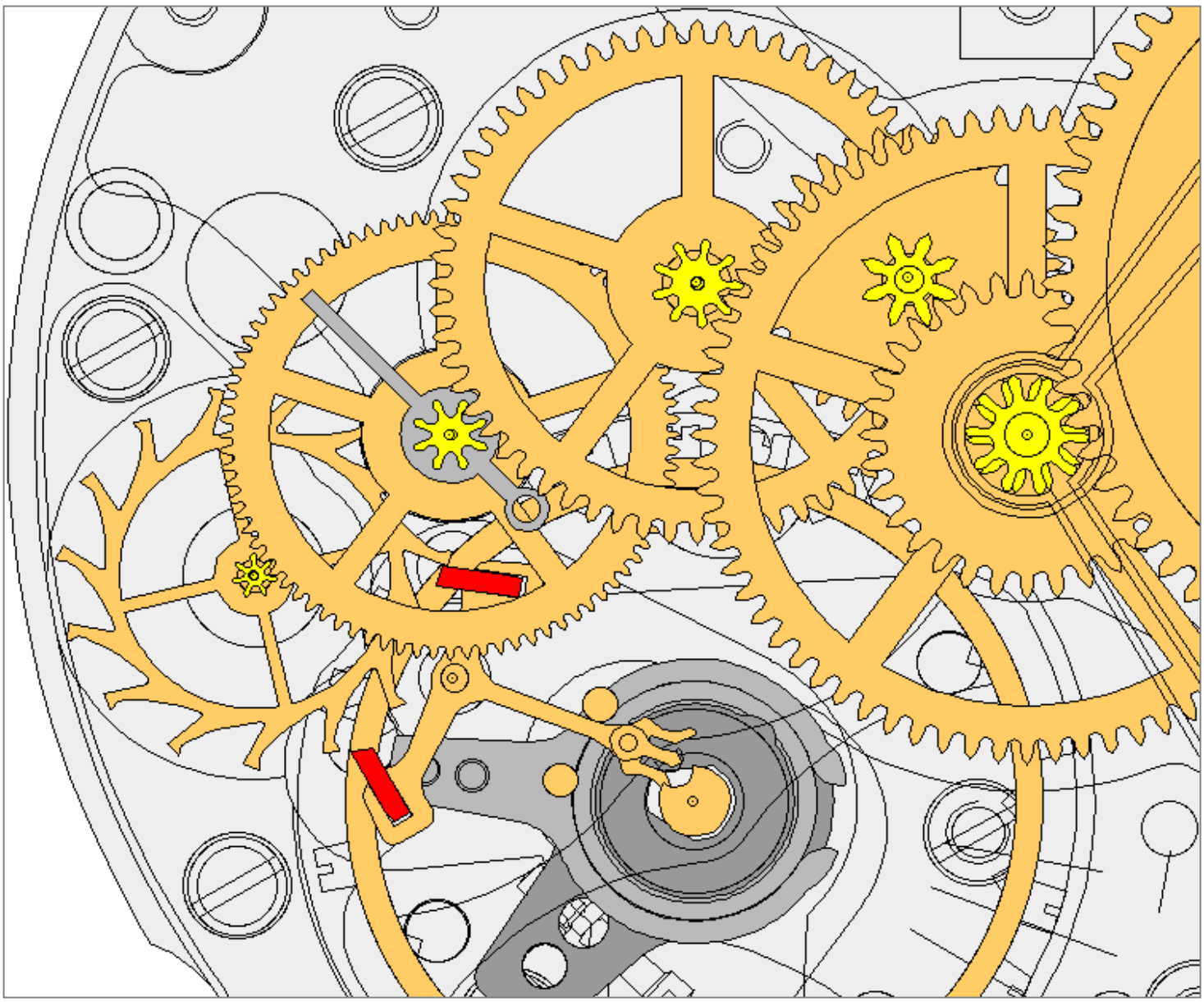
silux

The Multi-Body Dynamic Simulation Program

Tutorial (English Edition)



DOC2120_E

www.silux.com

silux ag · CH-6317 Zug-Oberwil · Switzerland

1 Documentation / Overview

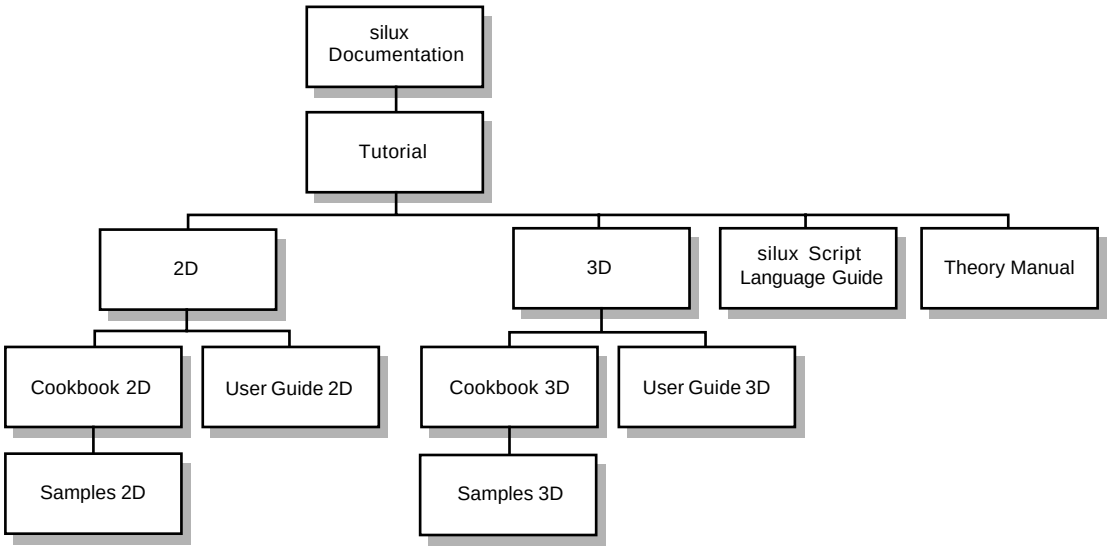


Abb. 1: (Note: The Theory Manual is in preparation and will be separately distributed - 1998)

Tutorial

Introduction to the program. This section contains background information about simulation and model building in general. All aspects of the program, its purpose, possibilities and potential are covered in this section. The tutorial allows a first look at the wealth of tools, functions and dialog boxes. After reading the tutorial, you will be able to recreate and execute the models and simulations that are explained in the cookbook.

User Guide

This is the central handbook and reference for the silux program. The handbook explains all aspects of using the program, completely and in detail. The User's Guide is generally not read in its entirety, but is primarily used as a reference if problems or uncertainties arise.

silux script Language Guide

The silux script language is fully explained in this handbook. The wide use of the command language, from the various macro uses to the language in the control systems and in

optimization tasks is covered. The commands are explained systematically and are ordered by subject.

Cookbook

The cookbook is a collection of examples whose creation is described entirely and in great detail. The actions that you need to take are listed in detail, step-by-step. The emphasis here is on presenting a chronological list of the commands and mouse clicks that you need to execute in order to produce a fully-functional model.

Samples

A collection of examples accompanied by explanations of the physics behind the models. The emphasis of this handbook is not on a detailed description of how to proceed with every action, but rather aims to lead the more experienced user through more demanding models. Many of these models are ”equipped” with background material such as reports, pictures and video clips. The completed examples can be loaded either as models, so-called TEXT files or as macros.

Theory Manual (in preparation - 1998)

The theoretical background material to the program will be provided here for advanced, demanding users. Some of the models presented in the Samples handbook will be used here as comparative references for corresponding analytic or empirical answers.

2 The Recommended Sequence

- Read this introduction (*Tutorial*)
- Read the first chapter in the *silux Script Language Guide*
- Try to understand a simple example from the *Cookbook*. The *Cookbook* is meant to explain a given example completely and step-by-step.
- If anything about the program is unclear, use the *User' Manual* as a reference, or esp. the Annex to the User Manual.
- If anything about the silux Script is unclear, use the *silux script Language Guide* as a reference.
- Try to operate the models discussed in the Samples section by loading the completed models and using the *Samples* Manual as an instruction guide.
- Try to create your own model and then try to execute simulations with it.

If, after studying this introduction, everything seems confused or even chaotic, if you don't understand the relationship between all of the different details approximately as shown in the following figure, then follow the indicated route through the highlighted stations as a guide.

- The highlighted boxes are the essential stations of the program.
- Everything else is extra.
- The extras can be added one by one after the essentials are clear.

The fundamental work routine that you need to remember is:

- Draw something (The drawing is produced from so-called raw data)
- Process raw data into objects (-> build physical objects)
- Set the model or simulation in motion (-> simulate)

3 Start Here

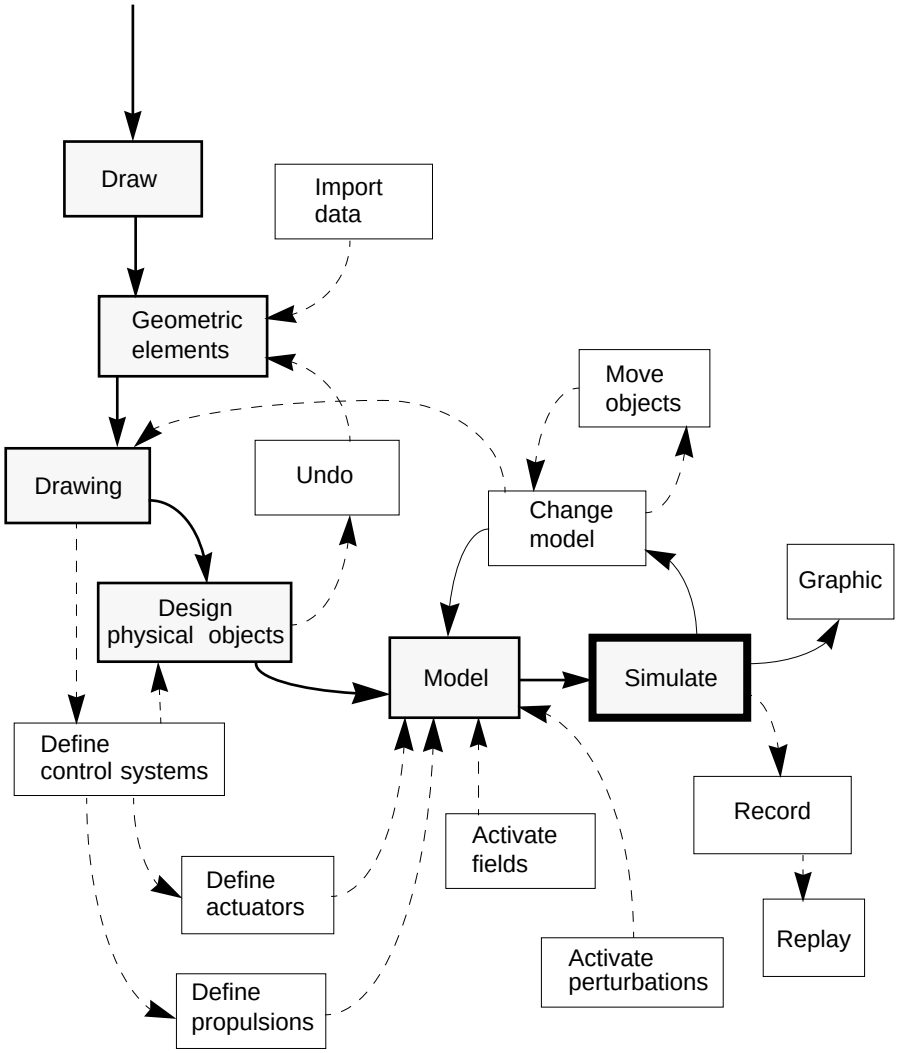


Abb. 2: Working with the program schematically — the direct way to your goal.

4 The Vision

We have a vision.

- We want to be able to simulate on the computer any machine that anyone can dream up.
- We want to be able to ”realize” any idea that belongs in the broadest sense to the field of mechanics.
- We want to be able to check the soundness of any piece of equipment that we have in mind on the computer.
- We want to be able to build virtual machines and to see how they work on the screen. Watch how they operate.
- We want to be able to carry out measurements on these virtual machines while they’re running.
- We want to be able to analyze these machines that exist, so far, only in our minds.
- We want to be able to implant autonomous controls into these machines. We want to see how a machine responds to given controls and, in this way, to be able to optimize machine management.
- We also want to be able to control these machines manually with a corresponding interface. We want to be able to build simulators.

We have a vision

- We want to encourage everybody to try something new, to take a risk, to travel new roads.
- We want to provide a tool that makes the testing of new ideas, especially these that are risky, expensive or extremely difficult to explore otherwise, easily feasible.
- We want people to charge after the realization of new ideas boldly. With silux no one will lose money, or "face", but will become experienced and proficient.

We have a vision

5 What is silux ?

This is the basic idea behind the program:

- Draw a machine
- When you're finished drawing, put the machine into action.
- Watch to see if and how the machine works.

With "machine" we mean any mechanical system and any physics experiment in the field of mechanics. We mean experiments from introductory physics classes as well as what we usually call machines. silux covers the whole range from classic physics experiments to complex machines with thousands of parts.

The silux program is a simulation program. silux models problems that belong to the field of mechanics in the widest sense. silux lets you solve problems realistically. By "realistic" we don't just mean that the program "seems real", but rather that the results delivered by silux match reality completely. The machine moves on the screen just as it would if it really existed off-screen.

You don't have to tell silux how a mechanical system that you've drawn moves or works. silux knows that already because it contains mechanical physics. The program will tell you how your machine moves.

So, as a spreadsheet program in the field of administrative simulations, silux allows you to wonder

- What if?

silux allows you to carry out simulations of technical tasks. With the silux program you can also ask the questions "What happens with my machine when I change this or any part?" "What happens if I let other factors influence the machine?" "What happens at the end if I change the conditions at the beginning?" "How does the machine react if I increase a part's size or the friction acting on it?" And so on....

6 silux, a Strategic Product

If cleverly marketed, a computer program that allowed you to type important meetings, lunchdates, vacations, etc. in software form into a computer instead of writing it into a leather agenda might sell. But no software developer will ever become a market ”giant” with such a product. Using this hard- or software will not give anyone a distinctive advantage over his or her competition. You would be doing the same thing as the guy with the leather agenda, just differently, more modernly and possibly in a more entertaining way. The new method may even be a few seconds quicker...but so what?

The same goes for the creator of a computer game. A word-processing program also belongs in this category. Even if a software developer produced a word-processing program with completely new features, like speech-recognition and direct speech-entry, you would not achieve a pronounced strategic advantage over the competition who is, as before, entering data with his or her fingers, even though such a program would likely be very marketable. In the end, the important element is the content, not the way things are executed. The user can probably achieve a certain improved efficiency, nobody wants to dispute that, but will his or her products be improved because of it?

There is another category of computer programs though and the people that own them enjoy a decisive advantage. There are programs that help the user to make discoveries that wouldn’t have been so straight-forward without the relevant program. There are programs that allow for the combination of discoveries with an insight that wouldn’t be possible without the program. There are programs that give the people that own them a decisive advantage over those that do not own or have access to these programs. There are programs that give the people that own them power.

There are, however, very few programs like this.

silux is a strategic product that gives you power.

7 Knowledge vs. a flood of information

The average computer-user will be plagued by an immense flood of information in the very near future. Access to this flood of information will be common property very soon. Going to the magazine stand is already a nightmare. You stand in front of walls of magazines that are tightly bunched together because there's no space: when looking through them you look like an archiver rummaging through the card catalog in the Library of Congress.

The worst part of it is that this flood of information is being accompanied by a noticeable loss of quality and enormous redundancy. Finding the best is becoming more and more difficult.

The merger of the television set and the home computer is imminent. Access to the Internet will soon be possible from every living room.

If you take a look at the sorts of computer software being offered, you notice that there are primarily two types of computer programs that promise us a shining future. The first type is the software that offers access to ever more information. The second type includes all programs that are intended to manage this flood of information.

At some point, everybody who has been dragged into this mire will ask the question "What is all of this worth to me?" People want to show us that knowledge is power and that knowledge is the hoarding of information. While the first part of that sentence is certainly true, we should seriously question whether the amount of information necessarily has anything to do with knowledge. People want to tell us that simply collecting information is "knowledge."

But "knowledge" has more to do with processing information. Knowledge is what you make of it. "Knowledge" is the conversion of information. "Knowledge" is what you call the inferences, discoveries and deductions that are drawn out of information.

Right now, a development that is opening up a divide between information communication and information processing is taking place. We have an inflation of information and processing cannot keep up. We need a new category of computer programs that helps us not only to collect and take note of this information, but also to process it. For us, the term "process" refers to capabilities similar to intelligence. We need tools that can bring us further with capabilities that we didn't have before. These tools should not only take over the work that we have so far managed in different ways. These tools should not just make work more efficient. Every software store is full of computer programs that offer their services to do things that we have been able to do for a long time in different ways.

We need tools that broaden and complement our abilities and open new horizons. These tools need to be able to take on tasks that previously could not be done. These programs should provide answers for questions that couldn't be solved before. These tools should make it possible for us to solve problems with methods that we hadn't even thought of before. These

tools should have their own, if specialized, intelligence. They shouldn't be programs that pretend to "know" everything. But the abilities that they offer should be unique.

silux is this new type of software..

8 From School Laboratories...

Children and young people are curious by nature. You can fascinate practically anybody with a lively, clear and experimental communication of knowledge. This type of communication was and is, however, resource-draining, time-consuming and expensive.

We want to provide interested young scientists or inventors or the simply curious with a program that they can use to try out all the physics and other scientific experiments that they could otherwise only play with in their minds. Not every child has access to a laboratory. Not every child can buy equipment for experiments. It doesn't make any sense either for every child to buy laboratory hardware. Many experiments can't be experienced in a laboratory anyway. The silux program is meant to turn thought-experiments into action.

With the silux program, we want to give every child or youth access to a laboratory in which he or she can try out all the mechanics experiments that would otherwise require a well-equipped physics lab. With silux, all kids can try out experiments that wouldn't even be possible in a Physics lab because they're too dangerous, too expensive, much too big, or much too small or simply not available.

- How far can you throw a stone on the moon?
- What happens when a comet of a certain size slams into the Earth?
- How fast will a rock go if you drop it from the top of the Eiffel Tower?
- What will the trajectory of something thrown from the space shuttle towards the Earth look like?

Clearly, some of these examples can be calculated pretty easily with a pencil and a piece of paper. But with silux, you can **see, experience and understand** them.

silux allows young people to play around with simple things like

- letting a rock fall
- letting something slide down an inclined surface
- letting a ball bounce

but can also solve trickier problems like

- constructing a marble game
- building a clock
- building a robot

That the child or youth isn't being fooled easily is a fact. The program isn't creating any illusions. No "canned goods" in the form of pre-prepared answers are presented, no pre-fabricated hints are offered. Pre-calculated simulation data or animation is never used in the

calculations. The program always calculates the corresponding physical answer in every possible detail ”ad hoc”, based on the prevailing situation, as far as it’s possible. The program calculates ”live”. The program simulates current reality.

Of course, everything discussed above also applies to all silux simulations, including those for industrial and scientific calculations.

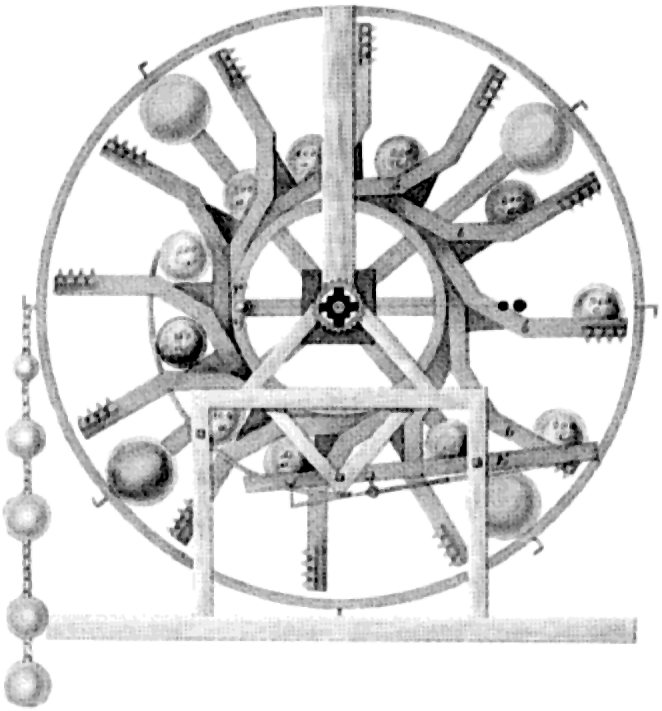


Abb. 3: Perpetuum Mobile

9 ...To Industrial and Scientific Tools:

A Scenario

A Swiss firm built the first aerial tramway with a two-level cabin. In this field, security regulations are as stringent as those involved in airplane production. The whole system must be tested for all possible incidents and safety must be assured in all cases. A possible scenario is a emergency stop directly over a cable pylon. In this special case, rough estimates show that the suspension of the cabin will collide with the so-called "saddle" of the pylon. The whole construction of the gondola has to be checked with a finite element program, also for this special case. The problem is that we don't know magnitude of the force or the duration over which it effects the aerial tramway upon impact. At this point, silux comes into the picture.

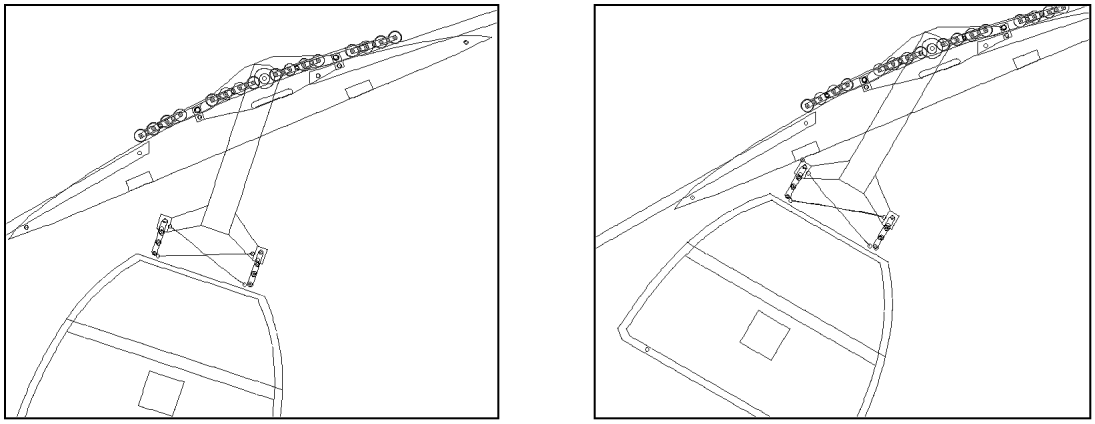


Abb. 4: The aerial tramway during an emergency stop directly over a pylon

The program simulates not only the dynamic relationships between all the individual parts and of the system as a whole in every detail, but also allows for the analytical investigation of specific forces and of other physical measurements.

If the aerial tramway is defined on the silux system, we can carry out all the experiments that are relevant to questions of safety. We can let the aerial tramway go down over the cable and the saddle and then instigate an emergency stop at a certain point. We'll observe what happens. For an examination of the already-mentioned force of impact, we'll fasten a sensor to the planned point of impact on the aerial tramway. Sensors in the silux program work exactly as they do in reality. We can assign this sensor the task of measuring the force acting on it

during impact and sending us a corresponding signal.

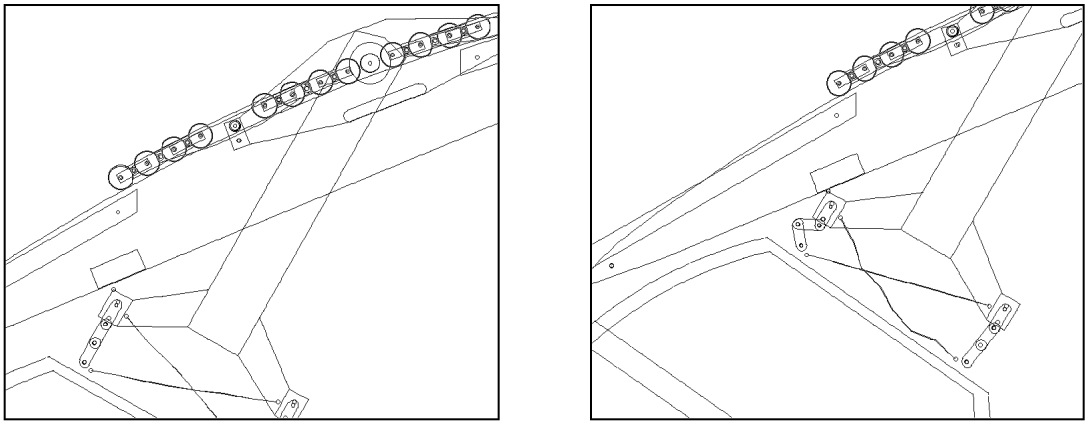


Abb. 5: The impact of the aerial tramway on the saddle. The small ball at the point of collision is the sensor. Notice that details such as the movement of the cable is also depicted.

We can take the signal of the sensor and immediately direct it to a graph that records the duration of the force.

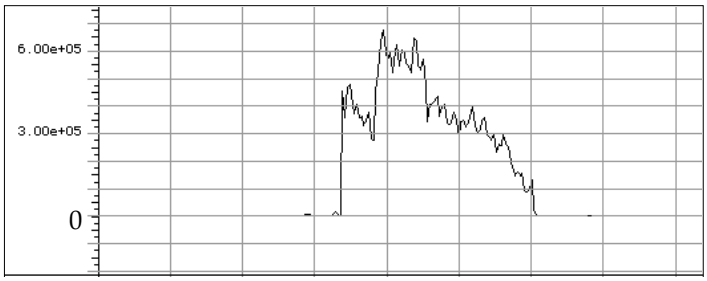


Abb. 6: The sensor's signal: Force as a function of time.

These measurements can be played through in any desired variation and combination of influence sizes. In this way you can test the scenario under different cabin-capacities (from empty to full), different gondola speeds and different breaking procedures. In the worst imaginable case, you return, in the end, to the finite element calculation to test the construction of the gondola.

10 Simulation

A simulation is a mathematical depiction of events that really happen or could really occur in nature. The concept ”simulation” is tied to the concept ”model”. A model is a description and a representation of certain concrete or abstract objects. A simulation is a representation of the behavior of the model under certain outside influences. One speaks of a simulation when he or she exposes a model to a certain environment and watches how it reacts.

A model is generally something that is comprehensible in itself. A model has such and such parts, the individual parts have such and such properties and are connected in this and that way and so on. The model may also be relatively complicated and consist possibly of thousands of complex parts. However, the description of the model is, in the end, something that is to be managed within a closed framework.

It is different with a simulation. A simulation of a certain model is generally a case of unending possibilities. A steel ball is a simple model. There are though a number of ways that this ball could roll across an uneven floor.

Experimentation, observation and theorization have been the classical methods of investigating scientific or technological problems for centuries. In recent times, the simulation has appeared. The simulation is in the process of becoming a new fundamental base of science. The computer simulation has become, along with the experiment and theory, the third pillar supporting research in the natural sciences. Specialized disciplines in which computer simulations play a primary role have developed in a number of classical subject areas of the natural sciences.

The procedure of examining a problem with simulation has certain advantages over experimental research. A simulation can be repeated as many times as is necessary, under identical or purposefully altered conditions. The influences of certain changes can be observed selectively, one at a time. Entire sets of solutions for groups of very similar experiments that are differentiated only a bit in regard to beginning conditions can be researched. With these methods we can discover and research new system properties that, with purely experimental methods, would only be discovered through a great exertion of time and money. A simulation can be stopped at any point and the condition of the model at that time can be observed precisely. An experiment that goes very quickly in reality can be carried through at any pace with a simulation. Natural events, that occur only once in reality, can be repeated as often as you want with a simulation.

The first division of the concept ”simulation” is divided into the following parts:

- Discrete simulations
- Continuous simulations

Discrete simulations are often associated with statistical problems. This type of method is

used with one-time events that occur with a certain probability or regularity. Problems that belong to this category include airspace monitoring, general traffic problems, stock-market speculation, problems involving lines (queues), etc.

Continuous simulations are based on mathematically more-exact principles. They deal with problems that take place under known rules in time and space. The simulation of the trajectory of a rocket is an example.

We differentiate more over between two classes of models:

- Deterministic
- Chaotic

Deterministic models are predictable, theoretically (entirely) calculable models. Chaotic models have such properties that an extremely small change in the conditions at the beginning have an extremely large effect (deviation) on the behavior of the model after a certain time.

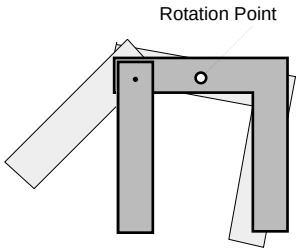


Abb. 7: Chaotic Pendulum

A simple example of a chaotic model is the so-called chaotic pendulum. If, under otherwise identical conditions, the force of the initial push is increased from, say, 10.000000 to 10.000001 Newton, then the movement of the pendulum will appear exactly the same to the eye. After a relatively short time though, the pendulum movements will be so different that they no longer have anything to do with each other. Chaotic systems are, then, only predictable and calculable for a certain amount of time. This does not represent a fault of the program, but rather lies in the truest sense in the nature of the thing. Maybe the best known example of a system that behaves chaotically is the weather and its prediction.

With the silux program you can execute comparative chaos experiments very well.

11 Methods of Simulation

(For readers that are familiar with higher mathematics)

The developers of simulation-programs might use one of many different methods of calculation for simulation techniques, depending on what tasks the program is supposed to perform.

The most important methods of simulation in the fields of the physical- and engineering sciences are:

- Finite Elements (Continuous simulation method)
- Finite Differences (Continuous simulation method)
- Monte Carlo (Discrete simulation method)

Finite Elements

The finite elements method is generally used to solve stationary or time-dependent field problems. From a differential equation that describes the physical facts and certain boundary conditions, you get a so-called boundary value problem. Mathematical physics recognizes different typical combinations of differential equations and boundary conditions that allow you to describe practical problems of technology.

Describing a problem with finite elements generally leads to a set with n equations and n unknowns or to the equivalent, a correspondingly large matrix that needs to be inverted.

The procedure with finite elements is also called an implicit algorithm procedure because the user obtains the solution to a complex physical problem through the solution of a boundary value problem by matrix calculation.

The method of finite elements is particularly good for static problems, equilibrium problems, stationary solutions for dynamic problems, determination of eigen-oscillations and eigen-frequencies, etc.

Finite Differences

Equations of motion are solved explicitly with the method of finite differences. Starting from certain conditions (beginning conditions), with a complete set of physical values and a specific point of time (t), the state of the entire system at time (t+dt) can be calculated. The chosen time-step (dt) is so small that the physical characteristics of the model can, with a good conscience, be regarded as practically constant during that time. On a laboratory scale, this time-step generally moves between a micro-second and a milli-second. Thousands, sometimes even millions of calculation-cycles are worked through during a simulation using finite differences. After each calculation-cycle, you have a new physical condition of the system.

The method of finite differences is suitable for dynamic or highly-dynamic problems, in which all or most of the parts are in movement. In the case of dynamic problems that have no equilibrium solution, there is no alternative to finite differences.

Monte Carlo

Monte Carlo is a statistical method of calculation that is generally preferred in simulation technology when an unpredictable sequence of events might come into play. With this method, you often make calculations based on probabilities that predict the occurrence of individual events and, by using a large number of examples, try to produce the best possible statistic result for the desired outcome.

Examples that belong to this branch of simulation include calculations dealing with people, traffic or flow of goods calculations, hit probability in defense technology, and so on.

Note

The silux program belongs to the class of continuous simulations and calculates according to the method of finite differences.

12 Simulation / How it works

The silux program works on the principle of small steps (finite differences). At the beginning of a simulation a model is in a specific condition. Each object has a certain position, a certain velocity, a certain spin and so on. Each physical variable has a specific value and the set of all variables and their values describe the model completely. The description of this condition is valid for a certain period of time, for the time being, the period of time before the simulation is started. The state of the model is therefore connected to a point of time. At each point of time (t) the model exhibits a specific geometric condition and a specific physical situation. Thanks to this complete description at time (t) we are able to predict the state of the model a short time step later (dt).

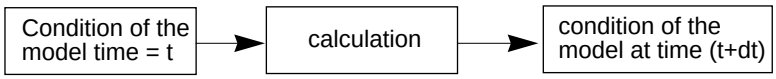


Abb. 8: Calculation step

If we then examine the model at the new point of time (t+dt), we'll notice that the individual parts of the model have moved a bit in location. Correspondingly, the values of certain physical variables will also have changed a bit. The state of the model has changed. This new condition is now a complete description of the model that allows us to begin the whole procedure again.

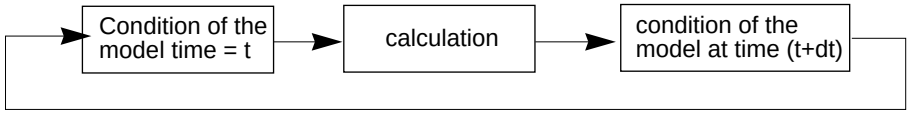


Abb. 9: Iteration

We repeat the cycle with the new state of the model and the new set of physical values. And we do this over and over (iteration). We move in increments of small time steps from one point of time to the next and in this way work our way forward in the time-scale. Each new point of time gives us a new picture of the model. We can see that on the screen. We see a film. The film probably runs in slow motion if the model is somewhat complicated, but we see how the model develops with the passing of time.

13 The Results of Simulations

The result of a simulation is not a numerical value or a formula but generally a great mountain of numbers. Processing this result can generally still only be done in graphic form. The possibilities for representing the results in graphic form is then as important as the simulation itself in a simulation program. In the silux program, the graphics that you ask for are processed and shown simultaneously with the simulation.

All physical aspects of each object can be represented over the course of time and in the desired form on the screen or can be saved onto a data carrier for later inspection.

The data flow can also be saved on an external disk so that it can be used at a later point with another suitable program.

14 Simulations and Computers

How fast does a computer have to be for you to be able to simulate with it?

Answer: As fast as possible.

Simulations, as they are carried out in silux, are among the most calculation-intensive procedures of all. Without fast computers, they're practically impossible. Super-computers were primarily developed for this purpose so that better and more detailed simulations could be produced. With each new generation of super-computers, simulation technology advances into new territories of knowledge that were previously inaccessible to simulation methods. The new generation of personal computers with RISC-technology meets the conditions necessary to run simulation techniques. In this sense, they offer the minimal performance that the user needs and expects to be able to work practically with simulation methods.

15 Time Steps

The size of the time step (dt) varies with the size or the extension of the model and with the size of the individual objects. The maximum permitted time step will be automatically calculated by the program.

For models on a laboratory scale, the time step generally moves on a scale between a millisecond and a microsecond.

When we set up a model that operates in the micron-scale, the time step is correspondingly smaller. The time step is also made smaller when we model very quick events. With ”very quick” we mean events that are as fast as or faster than the speed of sound in the relevant medium (300 - 3000 m/s).

The time step can also enlarged many times when we venture into astronomical areas. A time step on the scales of minutes is calculated for planetary movements in the solar system. For movements of whole galaxies, the time step increases up to a year.

If the maximum allowable time step is surpassed, the simulation will generally not work anymore. Not only are the calculations wrong, but the system will degenerate. The individual objects will begin to vibrate with an ever-escalating amplitude and the whole model will fly apart. The model explodes. The calculations cannot be kept under control any more. The system oscillations run rampant!

If the time step is smaller than the maximum allowable time step, there is really no effect on the quality of the simulation. It will just run more slowly. It can be mathematically derived that an iterative calculation such as those used will converge on the true exact solution if the time step lies under the prescribed limit.

As a user you shouldn’t be directly confronted with these time step problems because the system can manage the maximum time step and all of the simulation controls. The explanations given above are only for your information in case you wonder why the time step in a simulation is exactly a certain size and not smaller or larger.

16 Early Attempts with Finite Differences

Already in the 1920's, a certain Lewis Fry Richardson described a process of forecasting the weather with finite difference methods in his book *Weather Predictions by Numerical Processes*. He imagined a spherical, hollow area that was broken down into a grid. This area was a representation of the Earth's surface. In each square of the grid, there was a human "calculator" that accordingly covered that area of the surface. Each "calculator" estimated the physical values of phenomena like pressure and temperature for the next time period in his own square based on the values of these phenomena in the squares of his closest neighbors. In the center of the sphere was a director that made sure that the "calculators" were synchronized. Modern computer weather-forecasting and climate modeling programs work, in principle, exactly the same way. Today we know however that Mr. Richardson's individual calculators would have been too slow by many orders of magnitude to deliver a useful result within an adequate period of time, even if they'd calculated around the clock!

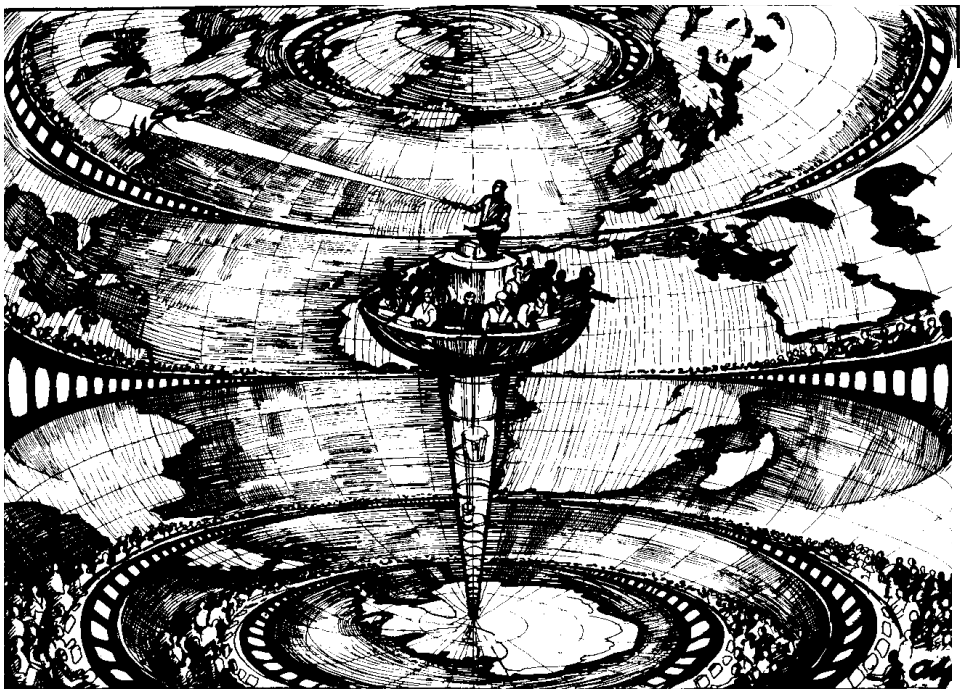


Abb. 10: Weather forecasting, L.F. Richardson style

17 The So-called Butterfly Effect

As scientists learned to work with simulation methods, it became obvious that the ”beginning conditions” of a simulation played an important role in regard to the precision of the results. The phrase ”beginning conditions” refers to the exact physical values that exist before the start of the simulation. Among many other variables, this includes the mass, the center of gravity, the geometry, the speed and so on of each individual part that plays a role in the simulation. These factors listed above are not yet very complicated to deal with. It is much more difficult when peripheral physical factors like friction or attenuation effects upon impact come into play: these effects are difficult to estimate in technology.

The beginning conditions play a particularly large, if not to say central, role in chaotic phenomena. We can return to weather prediction. As already mentioned, a very small change in the beginning conditions can or, perhaps better said, will cause the weather to move in an entirely different direction, first locally and then globally, after a certain period of time. This effect is not dependent on the absolute size of the change. If the change is made smaller, the shift in the weather will be slowed down, but it will still happen.

If, to take an example, you perform a weather prediction based on given beginning conditions, you will notice that the prediction corresponds with the actual weather for about three days, will tend to be right for another two days and will then turn in different directions. Afterwards, you can look more carefully to see where the causes of this error lie. You can manipulate the beginning conditions in tiny steps so that the weather prediction would have agreed with reality for up to 7 or 8 days (see also Reversed Engineering). A small change leads to different weather after a few days.

We want to refine the above with a thought experiment. Let’s look at the simulation program for weather prediction by itself, removed from the actual weather. With beginning conditions given, we’ll let the program make a hypothetical weather prediction for the next 180 days. Now we come to the point and will do the following. We will change the local beginning wind speed at a single place by an extremely small value, the smallest possible value that the computer can still work with. It’s on the scale of the breeze that is created when a butterfly flutters its wings. All of the other beginning conditions remain exactly the same. If we start the simulation again, we’ll see that perhaps four or five months later deviations occur that lead to the weather developing entirely differently after six months and a hurricane, that was not created in the first simulation, develops over the Atlantic.

Naturally we can’t check or carry out this ”butterfly effect” in reality. It can however be seen on the computer. This effect has nothing to do with the equations used or the mathematical models of weather-physics. It doesn’t have anything to do either with a problem that exists only on the computer. It has to do much more with the behavior and functions of nature. A butterfly influences the weather...

Is that really true or rather a philosophical problem?

Chaos-theorists tell us that it's true. Every living creature that moves on this Earth plays a part in the development of the weather. The weather isn't a force unto itself, we all create it ourselves.

18 Reverse Engineering

1. You take an existing solution to a problem.
2. You try various changes until you find a better solution.

Simply put, this is a discipline that has regained ground in recent times. We don't want to start from scratch with problem-solving, but rather want to use something that already exists in a similar form (perhaps from a competitor) as a starting-point. With this sort of procedure, a tool that allows you to examine an existing solution in great detail, to analyze it and finally to change it without great exertion is very useful. A tool that offers insight and makes possible changes that would otherwise be very hard to come by. A tool like silux.

1. You take an existing solution to a problem.
2. You transfer the solution to the silux simulation program.
3. You try various changes until you find a better, or optimized, solution.

19 Gear Technology

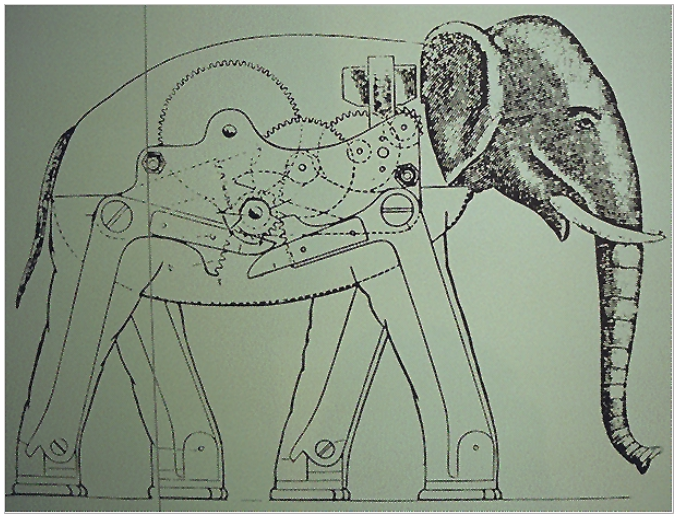


Abb. 11: Mechanical Elephant

Definition of Gears:

Gears are systems used to change or transfer movements or forces as well as torque. They consist of a minimum of three parts and one of these parts must be fixed as a stand.

- kinematic chain

-> If a part is chosen as the base =>

- Mechanism

-> One or more parts is driven =>

- Gears

Gear technology is an engineering science with its own theoretical base. It contains a system, calculation methods for gears of lower orders and instructions for the systematic analysis of existing gears.

A closed analytic representation of kinetic spatial gears is only possible in isolated cases.

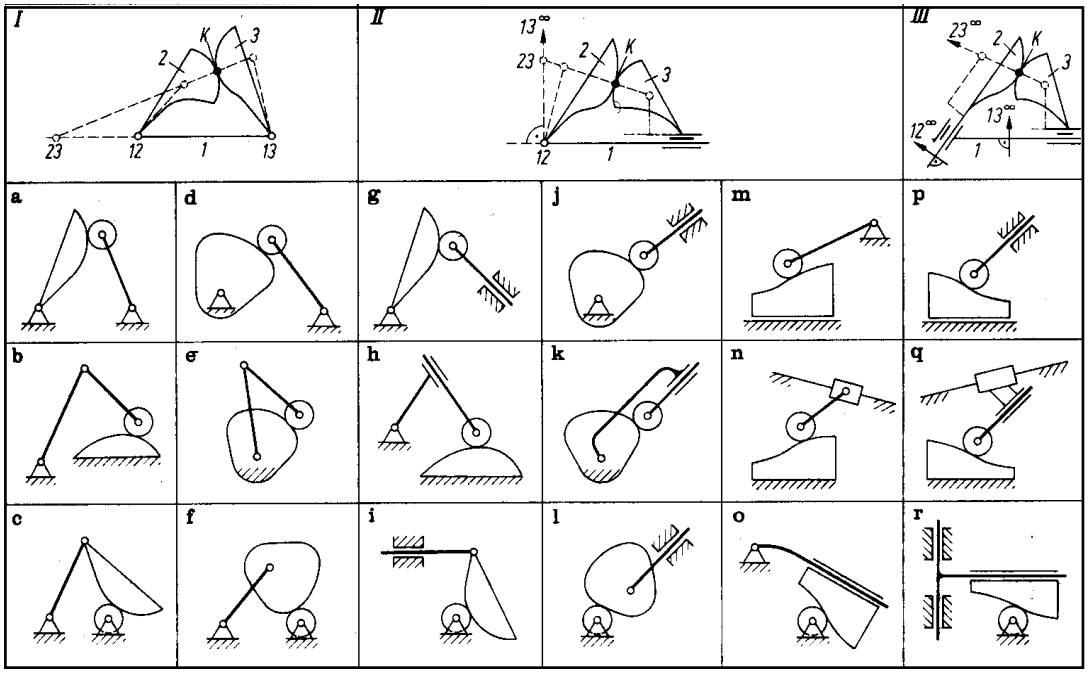


Abb. 12: Systematic three-part curve gears with turning and shifting joints.

There are various methods for the kinematic analysis of plane gears:

- Drawing-sequence-calculation method
- Loop iteration method
- Module method
- Kinetostatic analysis

In the last two or three decades, classical gear technology has been threatened in many fields by electronics and microprocessors. Often not because such solutions were better or simpler, but rather because they gave the product a better image (microprocessor Æ high-tech). Gear technology has been more frequently used in recent times. This is probably because of increasing pressure on prices: solutions based on gear technology are very inexpensive and have a very high operating security. People have also realized or observed again that with gear technology they can create processes of movement that could only otherwise be achieved with a complex exertion on control-systems.

The lack of a suitable tool of analysis often prevents engineers and constructors from using a solution based on gear technology. silux can really make a convincing contribution here. silux can simulate and show the dynamic behavior of even complex gears in a realistic and vivid way. Many gear-analyses are included among the examples that have been enclosed with the program. silux allows you to observe and gather features in gear-analysis that you would not have any access to through classical methods of analysis.

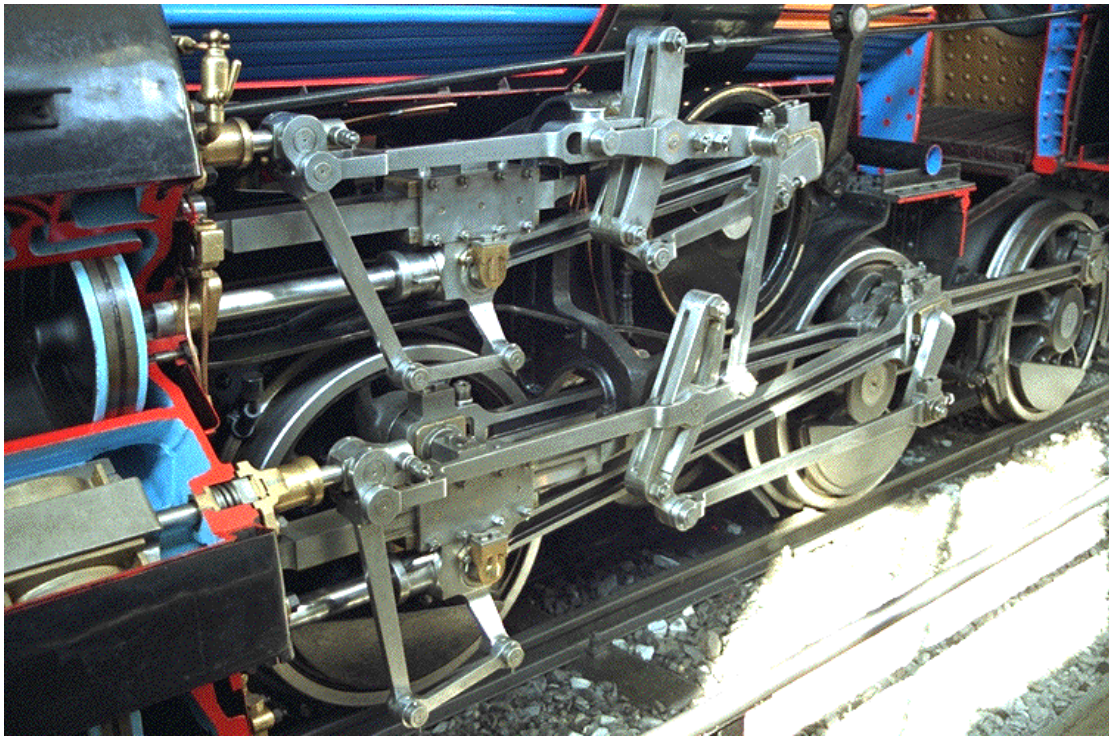


Abb. 13: Gear technology of yesterday (Verkehrshaus Lucerne, Switzerland)

20 From Automatons to Robots

If you combine silux's various abilities, you can build a robot in the true sense. A robot is a combination of machines, sensors and calculators. There are generally thought to be three essential components of a robot: a mechanical component, a logical or computer component and, as the link to the outer world, a sensor or perception component.

A robot is therefore a machine that can independently perceive outside events or stimuli with sensors, convert these perceptions through control engineering in the computer into reactions, and finally to execute appropriate actions with actuators. During the conversion of reactions to the desired movements, a field of mechanics called gear technology comes into play. (See Chapter 19)

silux possesses all these abilities. You can use silux to build a virtual machine that works with virtual controls. You can design and test controls for a machine that doesn't exist yet. You can see how these controls work out with the machine, even though up to now, there hasn't been either a machine or controls.

So you could, for example, present a possible Mars robot with the gravitational effects of that planet and a hypothetical Mars terrain and then watch how the robot moves and finds its way around in this foreign environment.

Industry robots will soon be further developed and will conquer areas where they haven't been used so far.

- Sensor technology and software will become more important in robot technology and will set the pace of further development.
- Robots will become cheaper, though perhaps not as quickly as we would like. The electronic and software components especially will become less expensive.
- Using robots will become simpler.

On these conditions, robots will also be used to a great extent in medium-sized businesses.

- Robots will have characteristics that can be referred to as improvisation abilities in the widest sense, though within certain boundaries at the beginning. Robots will become more flexible with autonomous behavior. The robot will be able to master changed conditions or small obstacles itself, or will at least be able to try.

A program like silux is a very helpful tool for promoting the development of this technology

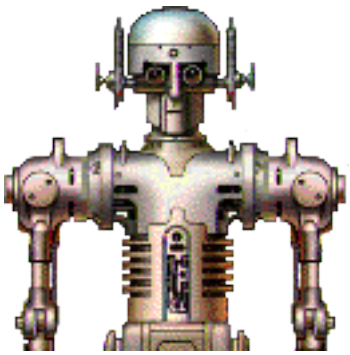


Abb. 14: A utopian robot. (From Isaac Asimov, The Ultimate Robot)

The three fundamental principles of robotics (according to Isaac Asimov)

- A robot must not hurt any human being or allow a human being to be hurt because of failure to act.
- A robot must follow the instruction of human beings, except when rule 1 would be broken.
- A robot must protect its own existence, except when rule 1 or 2 would be broken.

21 The Model

A model, in the sense of the silux program, is the entire collection of information that is necessary for a complete description of a mechanical system. It contains all the information that is necessary for a realistic physical simulation of the system.

- geometric description
- physical properties of the mechanical system
- physical properties of the environment
- mathematical description of the model controls and control systems
- silux programs for the program control, the model optimization, etc.

The silux program takes care of transforming your ideas, that is, putting them to use and simulating the models. The silux program knows mechanical physics. You define the properties of the model, silux will show you its behavior.

Defining a model generally proceeds as below:

1. A graphic definition

2. A physical definition

3. Possibly a definition of model controls or control systems (if they exist)

4. Possibly a definition of program controls (if they exist)

22 The Size of the Model

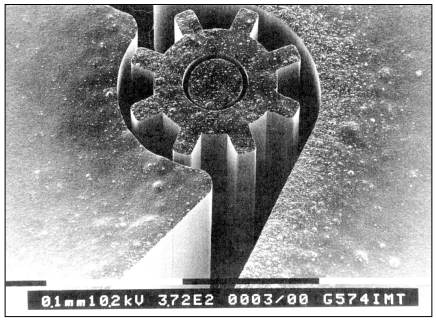


Abb. 15: Micro-Technology

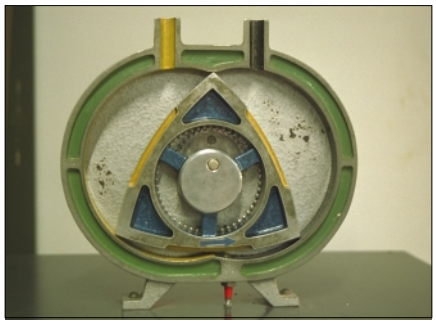


Abb. 16: Micro-Mechanics

Abb. 17: Large-Scale Installations

Abb. 18: Astronomical Calculations

23 Two Methods of Using Models

When you construct a mechanical system with silux and start a simulation, that is, set a model in motion, you can observe the behavior of the model, watch how the state of the model changes and have graphs displayed wherever the details interest you. If the model contains automatic controls, you can also follow the effects of these influences. Whenever you think it's necessary, you can stop the action, change something and possibly set the whole thing back in motion. Or maybe you're only interested in the final result, the state of the model after a certain time. You may want to compare the different end results that follow from using different variables. The emphasis of this method lies in

- Analysis

silux however also allows you to install interactive controls. You can manage these controls interactively through hardware components such the keyboard, the mouse and the joystick. You can determine yourself which actions are controlled or changed by what hardware component. You could for example, set it up so that when you hit the [v] key on the keyboard, a certain actuator moves out automatic programmed controls and then moves them away again when you push the [r] key. You can manage several different controls in the same model at the same time manually. The emphasis here is then on interactively controlling: we are building a

- Simulator

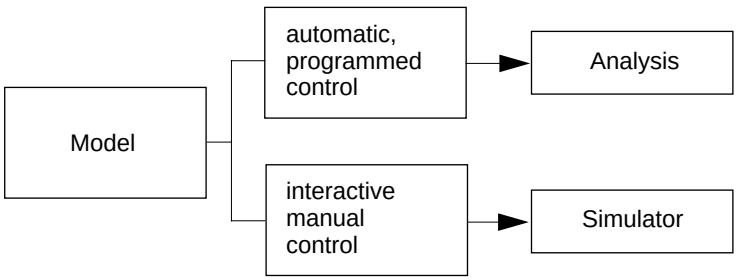


Abb. 19: Two Methods of Using Models

24 Model Building-Blocks / Overview

A model is generally based on a selection of the following building-blocks:

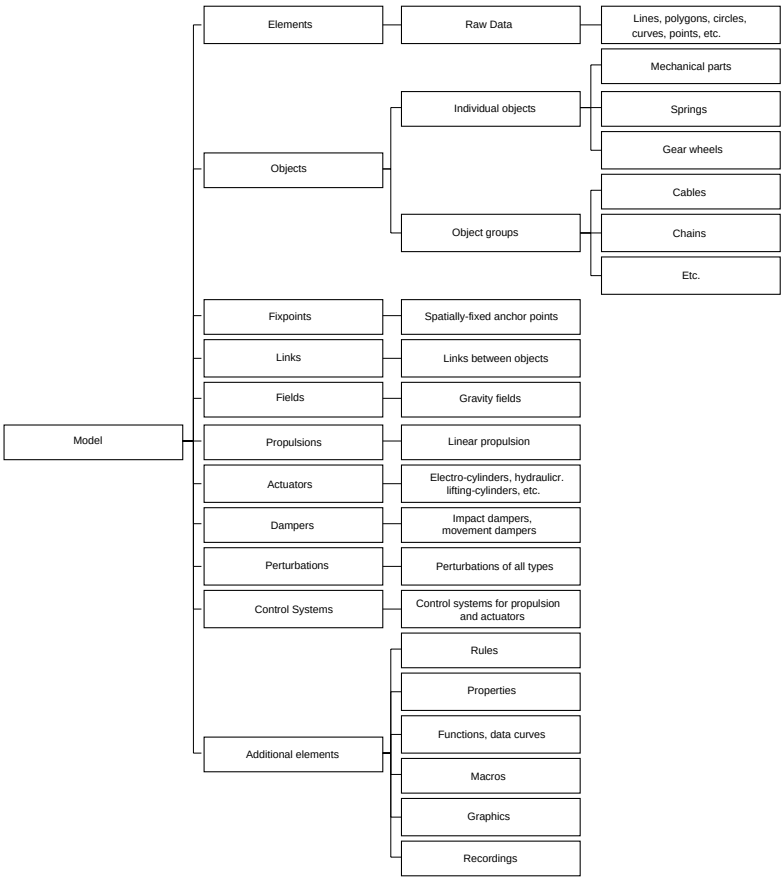


Abb. 20: Building-blocks of a Model

When they're used, these building-blocks belong to the corresponding model and, as such, are automatically saved and opened with the model. The most important building-blocks are:

Element

An element is a raw-data unit. The following element-types are available in silux:

- POINT
- LINE
- ARC
- CIRCLE
- RECTANGLE
- POLYGON
- POLYLINE

Elements are pure geometrical structures. Elements have no physical ”inner life” and remain undisturbed by outside physical influences. Elements can be grouped together to form element-groups. Elements and element-groups are often processed into objects. If you use the ”Drawing Toolbox” to draw elements on the silux screen, you are producing elements or raw data. Data that is imported from another program in an exchange format (DXF) is taken up as raw data by the program. Elements are generally drawn in a different color on the screen so that they can be differentiated from objects and other types of data.

Object

An object is a physical body. Objects can have any boundaries: under various circumstances it may consist of many single closed elements and element groups. Objects obey the physical laws that are valid throughout silux, and those that are additionally defined within a certain model. You can define the inner physical properties of an object extensively and individually. There are basically two classes of objects:

- Rigid objects with a solid exterior
- Complex objects with an internal grid-structure and an elastic outer shell.

In a standard problem, all or most of the objects are rigid bodies. You will probably process individual objects into complex bodies with inner grid-structures only rarely (only possible with silux 2D-PRO), in cases of specific interest, if at all. Complex bodies lead to a serious increase in calculation-expenditure.

Fixed Point (Fixpoint)

A fixed point is a mass-less dot that never changes its position with respect to the global coordinate system. Fixed points generally serve as anchors for physical objects. Objects are bound to fixed points with links. An object that is linked to a fixed point may move freely around the point of contact, unless other conditions restrict its movement.

Link

A link is a point that serves as a relative connection between two objects. The point of contact

has a specific position with regard to the local coordinate systems of both objects. Two objects connected in this way can both turn freely around this point, as long as that point is the only bilateral connection between them. If one of the two objects is a fixed point, then the connection is spatially fixed.
A link is always an elastic connection. The physical properties and the characteristics of a connection can be changed and fixed individually by the user within a wide parameter field.

The program provides a so-called torsion-link in addition to the normal link. A torsion-link exerts a torque that is proportional to the relative, mutual angle of rotation on the two joined objects

Propulsion

Propulsion is a function that must be joined to an object. Propulsion is a mass-less, mathematical thing that exerts a force or torque on the object: this propulsion can be managed with regards to the size of force, direction and control in many different ways. You can control propulsion individually and directly through the keyboard. Selected propulsions can also be controlled automatically resp. left to a control-system (see Control Systems). A single object may have multiple propulsions that can be controlled independently of each other. An example of a propulsion that produces a force is the thrust-propulsion (rocket motor). Examples of propulsions that create torque are electric or gasoline motors. A propulsion is then, in the silux program, something that we view as a ”black box”. We are not interested in how the force or torque has come to be, but rather concern ourselves with what control-variables to use and how big the force is, what direction it takes, etc. as a result.

Actuator

An actuator is a servo-motor or a correction element or an actuator sleeve or a linear motor. An actuator is something that, after the entry of an operating value or a signal, changes its length or angle of rotation. An actuator is a subassembly that can be created as a whole. If you call up the function for creating an actuator, you’ll have the opportunity to determine the properties of the actuator extensively in a dialog box. You can control the actuator individually and directly through the keyboard. You can also automate the control of the actuator by leaving it to a control-system (see Control Systems). Examples of actuators are the hydraulic cylinders of a dredge which control the placement of the shovel. An actuator in silux is again something that we view as a black box. We don’t care about how the powering of the actuator itself works (electric, hydraulic, pneumatic, etc.) but rather only which control variables go into the box and correspondingly which values of the length, (push-speed, the maximal force, etc.) that come out.

Damper

A damper is a mechanism that hinders relative movement between two objects by exerting a force that acts against the movement between the two objects.

The Effects of Some Model Building-Blocks and Basic Functions of the Program.

Building-Blocks	Effect / Function (vector)	Example
Linear propulsion	Force (time)	Rocket motor
Rotational propulsion	Torque (time)	Electro-motor
Actuator linear	Distance (time)	Electro-cylinder
Actuator rotative	Angle of rotation (time)	Stepping motor
Tension spring/ pressure spring	Force (distance)	Tension spring
Torsion spring	Torque (angle of rotation)	Torsion spring
Damper linear	Force (velocity)	Shock absorber
Field	Force (position)	Gravity field
Perturbation	Force (time, random)	Wind
Gear wheel	Angle of rotation (angle of rotation)	Gear wheel
Outside forces	Force (velocity, etc.)	Wind resistance

Tabelle 1: A Brief Morphology: Effects of some Model Building-Blocks

25 Abstraction Levels

In the following, we want to give an introductory view of how the individual objects that make up the models are themselves modeled. With how much detail and with what degree of correspondence to reality can the individual objects, links, actuators, dampers etc. be reproduced? Where are idealizations or simplifications made on these objects?

The following table will give you initial answers to these questions.

Object	Description	Abstractions	Notes
Object	Objects can be modeled very precisely. The boundaries of objects can modeled after their subject in the greatest detail. The resulting calculations of mass, center of gravity, etc., that take place in the background are as precise. Some limitations are however presented by the two-dimensional model. Sometimes you must represent a three-dimensional model in a two-dimensional idealized drawing because a more realistic translation isn't possible. The silux-2D program calculates in 2.5 dimensions and even allows for a simplified modeling of 3-D structures (see User Manual: the Object). The interaction between the objects is in any case only 2 dimensional.	Aside from the 2 dimensional limitations, there are no abstractions.	
Links	Links are dot-shaped connections between two objects. Links are axles in a mechanical sense. The mass of the axle and of the bearing are taken on by the two objects that are linked. The link simply defines the dot-formed point of rotation of the axle and the bearing. The bearing force and the friction in a link are controlled by the program and by the user. Despite the geometrical idealization, the bearing functions absolutely correctly in a physical sense.	In a geometric sense, the axle and the bearing are reduced to a point	A bearing can in principle be represented with circular axles and bearings instead of with a dot-shaped link. This representation does not however provide any physical advantage over a normal link.
Actuator (linear operating-cylinder)	Worldwide, there are thousands of different actuators. In regard to the outside form, an idealized actuator that can act on behalf of all possible actuators has been modeled in silux. In the silux program, the actuator is split up into an idealized geometric cover that consists of casings and cylinders and a programmed "inner-life". The "inner-life" or the exact function of an actuator can largely be defined by the user. You can also assign the actuator a mass. The idealized actuator works on the screen correctly in regards to the movements of the casing and cylinder. An actuator can be controlled manually or with a control-system.	Geometrically idealized casing and an idealized operating-cylinder. Functionally a construction kit of many different function units that the user can join to form entire functions.	An actuator could be geometrically and functionally related to a real model by building the actuator up from objects exactly and incorporating an invisible standard actuator for the function or programming.

Tabelle 2: Different objects and their implementation in silux.

Object	Description	Abstractions	Notes
Damper	Things stand similarly with dampers as with actuators. In regard to the outside form, one idealized damper that might stand in place for all dampers has been provided in silux. The ”inner-life” or the exact damping properties of the damper can be defined by the user.	Geometrically idealized case and idealized work-cylinder. Functionally a construction kit made up of different function elements that are joined together by the user to form an entire function.	A damper can also be geometrically and functionally related to a real model by building the damper up from objects exactly and then incorporating an invisible standard damper for the function.
Propulsion	Propulsion is a purely mathematical thing. It is represented in models with a symbol. Propulsion is coupled with objects. Each object can be outfitted with one or more different types of propulsion. In this sense, a distinction is also made between a physical shell (object) and a functional or mathematical part (propulsion). The functional ”inner-life”, the exact function of propulsion, can be extensively defined by the user.	A mathematical thing that is coupled to an object. Functionally, it is a construction kit of different function elements that are joined together by the user to form an entire function.	The true appearance of propulsion can largely be determined by the user by simply designing the object that the mathematical propulsion is linked to correspondingly
Helical Springs	Helical springs are mathematical things that are represented by symbolic spring figures. The springs change their appearance correspondingly when their length is altered. The user can define the functions and the physical properties of the springs in the corresponding dialog box him- or herself. Loose springs behave like physical objects. Springs that are linked loose their physical casing and become mathematical things.	Linked springs have no mass and are firmly bound at both ends to their partner objects.	
Torsion Springs	Torsion springs are mathematical things that are not graphically represented in the model. A torsion spring consists of two links. The first link is responsible for the axial fastening of the two objects that are joined by a torsion spring. This first link can also be assigned a frictional element. The second link, the so-called torsion link is responsible for the torsional function of the torsion spring itself.	A purely mathematical thing	
Cables	A cable is a chain of individual, cylinder-shaped objects that are bound to each other by elastic links. The physical behavior of cables largely corresponds to reality in every relationship. This is true for the local behavior along the cable as well as for the cable as a whole. There are very high-performance tools available to the user for the creation of cables. Collision control for cables functions as with objects automatically.	The inner construction of a cable is abstracted from. There is no inner cable construction of individual wires.	

Tabelle 2: Different objects and their implementation in silux.

26 Rigid Bodies / Elastic Bodies

The silux-2D program is available in 2 versions.

silux 2D-Standard

This silux version calculates with so-called rigid bodies. People speak of rigid bodies if the surface or the borders of the body cannot be deformed. In the case of two dimensions, the border of a rigid body is a solid geometric contour that doesn't change during the simulation.

The user can, of course, edit each body between two simulations and in this way change its form and appearance. You cannot however observe a deformation resulting from force-effects during a simulation with rigid bodies. A beam will never bend under pressure. This detailed solution is at any rate not of any interest in many problems. Users are generally interested in the dynamic behavior of the individual bodies as intact units in the entire system and in the forces between the bodies.

silux 2D-PRO

With the silux 2D-PRO version, you are additionally able to calculate with so-called inner grid-structure. This expanded version contains a complete finite element package. silux-PRO allows you to observe the elastic-plastic behavior of objects. This additional ability can be called into use for specific purposes by incorporating a grid-structure into in bodies of particular interest. A typical silux-PRO model is made up of a mixture of rigid bodies and those bodies with an internal grid-structure. The following properties can also additionally be observed during a simulation with silux-Pro.

- The elastic behavior of bodies
- Plastic deformations of bodies
- Inner tension distributions
- Shock waves

Note

Everything that follows in this handbook is equally true for both versions. All the abilities mentioned in this handbook have been implemented in both silux versions.

27 Program Environment - Overview

After you start the silux program and wait a few seconds, the screen will look like this:

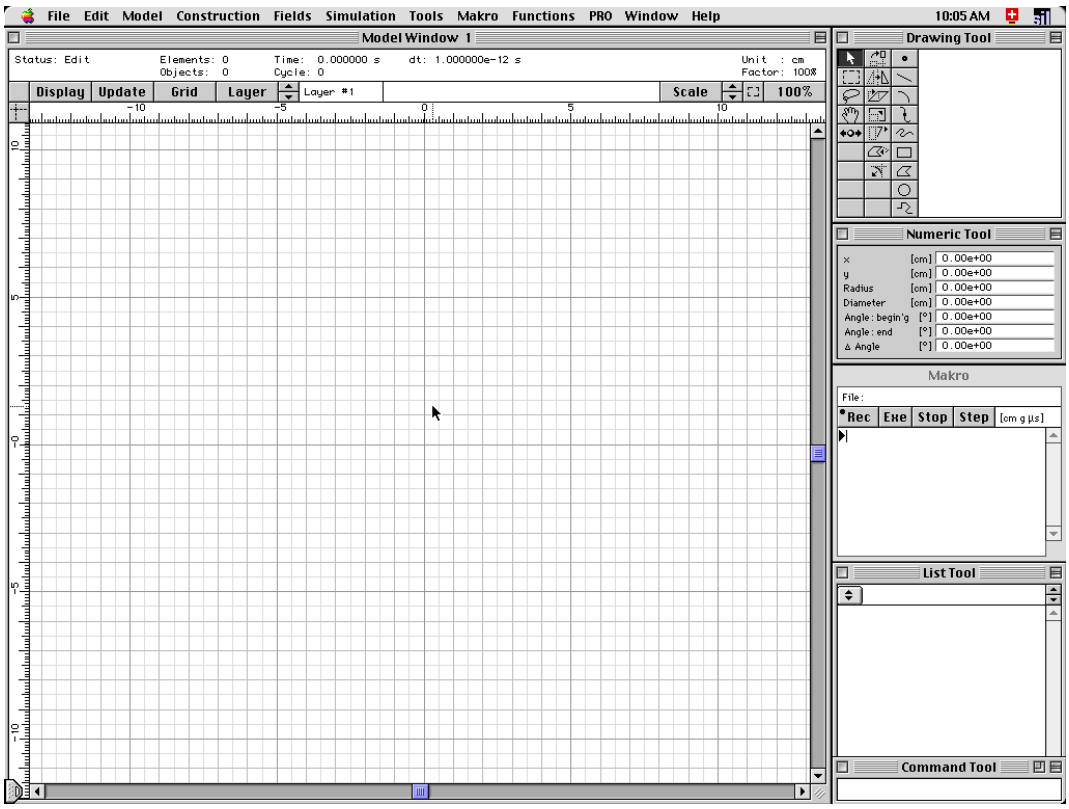


Abb. 21: Standard Screen

You can see one large and a few small windows. The large window is the true model-window. In the model-window, the model that is drawn, loaded, simulated, etc. is represented. All the windows along the length of the right-side of the screen are the so-called tool-windows.

Each tool-window is responsible for a specific function. The tool-windows can be opened and closed and replaced by other tool-windows (separately). Different sets of Tool-Windows will likely be called for at different stages of model development or implementation.

We differentiate, among others, between the following Tool-Windows:

- Drawing Tools
- Numeric Tool
- Material Tool
- List Tool
- Macro Window
- Command Window
- Graphic Window
- Information Window
- Replay Tools
- And so forth

28 Model Window

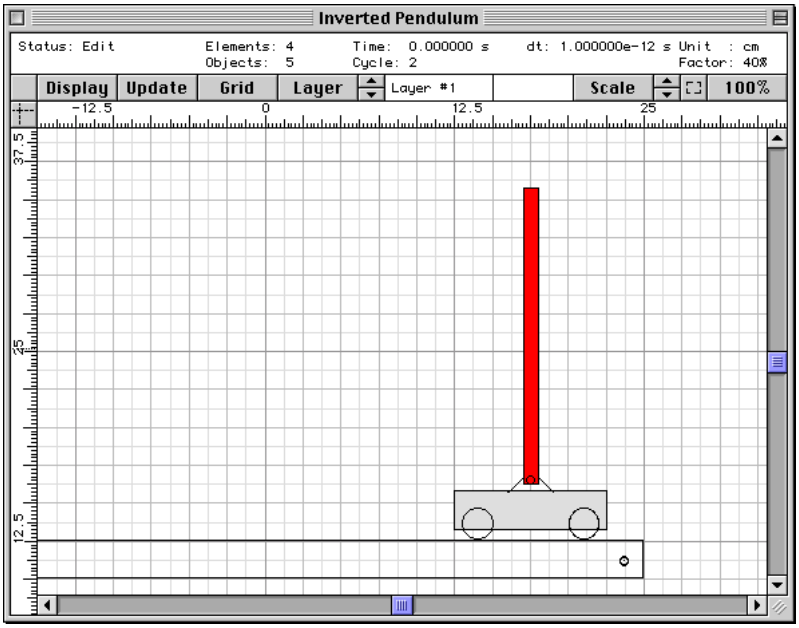


Abb. 22: Model Window

The model window is the carrier of a model. The following things happen in the model window:

- You define the model (draw, edit, change, modify, etc.).
- You simulate the model.

The model window is the graphic representation of the model. The representation itself can be carried out in different forms. You, as the user, can influence this. During a simulation, if the parts of the models move and change their positions, you will constantly be shown the most recent position of all the parts. Therefore you see on the screen how the model moves.

The definition and the simulations of the models take place then in the same window. You can switch back and forth between these two tasks interactively and without problems on the screen. A model is not simply static during a simulation. A model can be defined, the simulation is started, stopped after a time, some part of the model modified, and then the simulation will continue, taking the changes into account. The program is a laboratory.

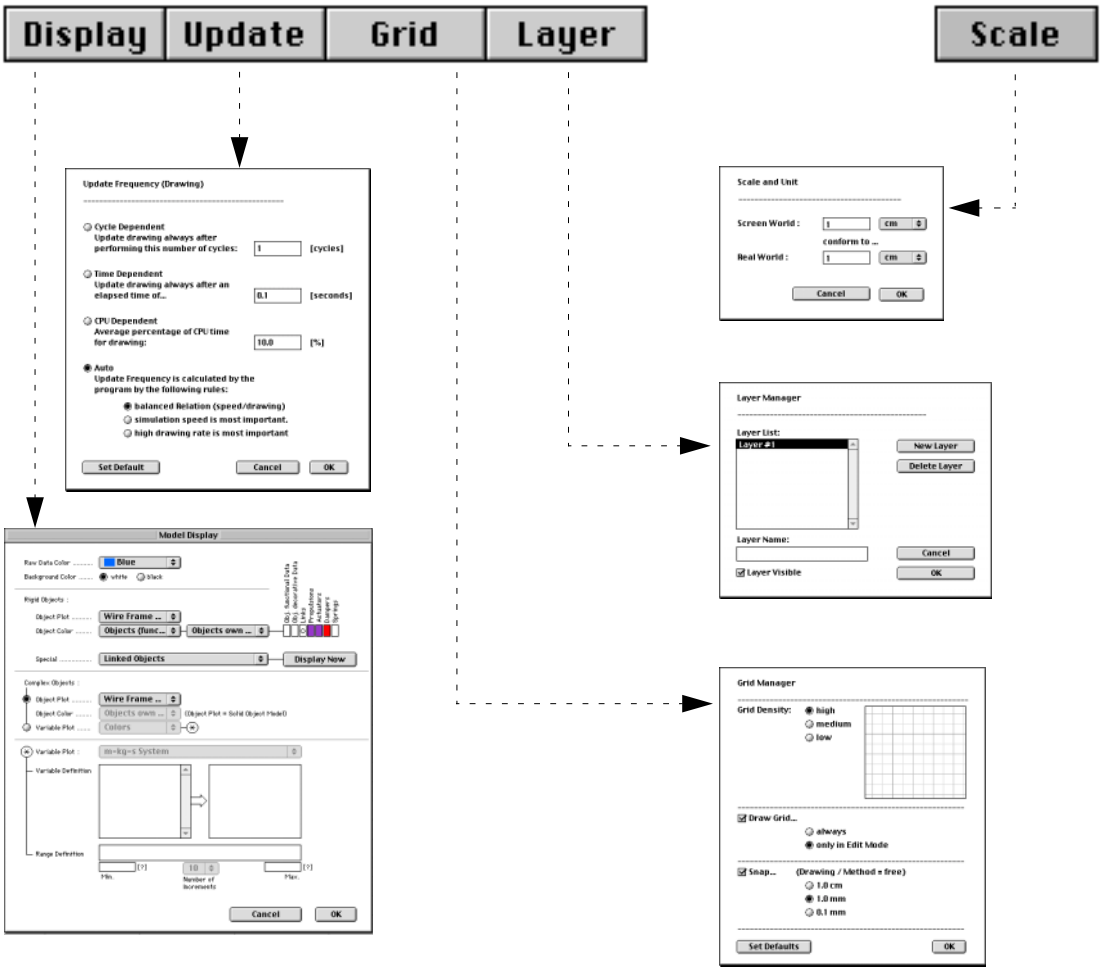


Abb. 23: The button bar in the Model Window

Various dialog boxes can be called up from the button bar in the Model Window:

- **Display**: Which set of data should be shown and how?
- **Update**: How often should be representation of the model be renewed on the model window?
- **Record**: Turning on and off the recording of a film.
- **Grid**: Should a grid be laid behind the model and, if so, how?
- **Layer**: Layer-management. The model can be built on different levels..
- **Scale**: Determining the scale.

29 Drawing Tool

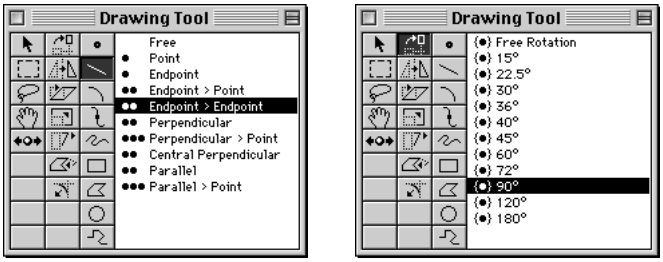


Abb. 24: Two different modes of the drawing toolbox.

The drawing toolbox contains the most important aids for the graphic and geometric definition of models. Within this tool-box we differentiate between three groups of functions, which are divided into three columns of buttons from right to left:

- Drawing elements (line, polygon, circle, etc.)
- Modifications of these elements (rotate, stretch, mirror, etc.)
- Additional operations (area-selection, shift drawing, etc.)

There is an additional section to the far right, next to the three button columns. Lists of the possible options or methods relevant to the chosen function are shown in this area as a "choice list". You can click on one of these choices and then carry out the chosen function exactly the way you want to.

Certain options require that you have already chosen certain selections so that the corresponding function and method can be carried out. For example, if you choose the line function you will be shown a list of methods with which a line can be defined. Before certain methods, you'll see one or more point-symbols. If you want to draw a line from one point to another, you'll have to have already chosen two points so that the system knows which points to connect to form a line. You see the two points in front of this drawing mode for this reason.

If a point is in parentheses, the pre-selection of this point or points is optional. So for example with turning: an already-selected point will be designated as the point of rotation. If no point has been selected then the rotation will take place around the point where the mouse has been placed.

For further information about the Drawing Tools, please see the*User Manual*

30 Numeric Tool

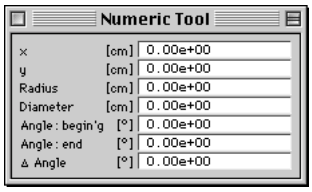


Abb. 25: The Numeric Tool box

The Numeric Tool performs two main tasks:

1. Numeric indication of position

- during the definition of an element
- during the selection of a point.

2. Changing the position of individually selected points through entered numbers.

During the definition of an element, the position and other geometric values such as the angle or the radius of a arc can be seen constantly in the Numeric Tools Window. You can concentrate on the Numeric Tools instead of on the scale while drawing.

When you click on and thereby select a point with the mouse, the exact position of the point is shown in the Numeric Tools box. By ”exact” we mean that the exact saved value of the corresponding point is shown, not just a conversion of screen-coordinates.

The coordinates displayed in the Numeric Tools box can be replaced by other values: as soon as these changes are made in the box, the corresponding changes will take place on the screen. In this way, initial or final angles or curve radii can be changed numerically, and so on.

It’s worth noticing that there are also points that cannot be changed numerically. For example, the intersection point of two lines can be selected and shown in the Numeric Tool box. Changing the coordinates of this intersection point is however not possible. The selected point of intersection can however be used again as in analytical geometry and, for example, be designated as the center of a circle.

31 Material Tool

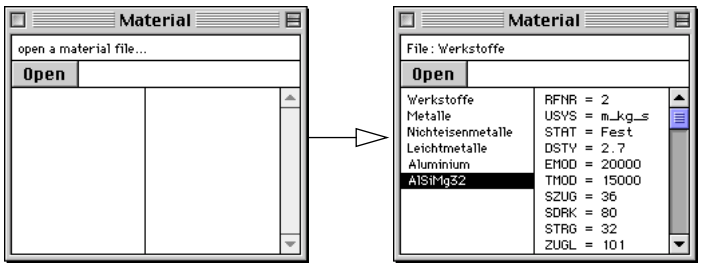


Abb. 26: Material Tool

The Material Tool or Material Window is your interface with the material-databases. Your material data can be sorted by the type of material and saved in as many material-files as you want. You can open all of these files with the same Material Tool, always one at a time.

There is a button in the upper part of the Material Window.

- Open

After clicking on the ”Open” button, the standard computer dialog box for the selection of a file will be shown. This dialog box allows you to open drives, folders and finally files that contain the material-data that you need. Typical material-files, to name just a few, are:

- Metals
- Synthetic materials
- Springs
- Cables
- And so on...

Certain material-databases are provided along with the program but can be added to by the user and expanded with the addition of new files. Material-databases are files in a special format.

For further information about the material-editor and the file-format of the material-files, please see: *User Manual - Anhang C*

32 List Tool

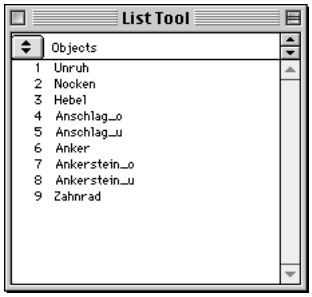


Abb. 27: List Tool

The List Tool is a platform that shows all sorts of different lists and allows you to select one of these list-elements by clicking on it. By double-clicking on a list-element, you will automatically open the corresponding dialog box so that you can choose the list-element.

The following lists can be called up through the List Tool with a pop-up menu:

- Elements
- Objects
- Fixpoints (Fixed points)
- Links
- Propulsions
- Dampers
- Actuators
- Control Systems
- And so on...

When, for example, the list of all objects is shown, the corresponding object can be selected by clicking on the appropriate list-element. The selection of the object occurs in parallel in the Model Window itself and the appropriate object will appear there with markings. When you do the procedure the other way around and choose the object in the model window by clicking on it, the object will then be selected in the List Tool. The Model Window and the List Tools are then two corresponding forms of representation of the same thing. The information is given once in graphic form and once in text or list form. Double-clicking on an object in the List Tool automatically opens the Edit Object Dialog box. The same thing happens in the Model Window when you double-click on an object.

33 Macro Window

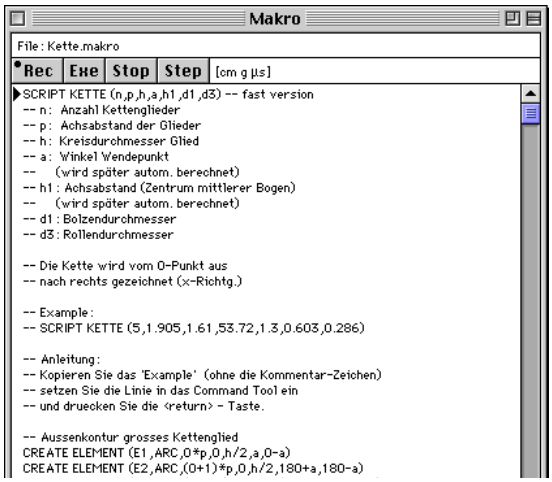

Makro
File : Kette.makro
Rec
Exe
Stop
Step
[cm g µs]
SCRIPT KETTE (n,p,h,a,h1 ,d1 ,d3) -- fast version
-- n: Anzahl Kettenglieder
-- p: Achsabstand der Glieder
-- h: Kreisdurchmesser Glied
-- a: Winkel Wendepunkt
-- (wird später autom. berechnet)
-- h1 : Achsabstand (Zentrum mittlerer Bogen)
-- (wird später autom. berechnet)
-- d1 : Bolzendurchmesser
-- d3 : Rollendurchmesser
-- Die Kette wird vom 0-Punkt aus
-- nach rechts gezeichnet (x-Richtg.)
-- Example :
-- SCRIPT KETTE (5,1.905,1.61 ,53.72,1.3,0.603,0.286)
-- Anleitung :
-- Kopieren Sie das 'Example' (ohne die Kommentar-Zeichen)
-- setzen Sie die Linie in das Command Tool ein
-- und druecken Sie die <return> - Taste.
-- Aussenkontur grosses Kettenglied
CREATE ELEMENT (E1 ,ARC,0*p,0,h/2,a,0-a)
CREATE ELEMENT (E2,ARC,(0+1)*p,0,h/2,180+a,180-a)

Abb. 28: Macro Window

A macro is a description of a procedure or a series of procedures taken by the user in a language developed especially for this task. The language is called silux script. You can either let the system automatically record macros for you or you can write them yourself with a so-called macro-editor. Macros help us to automate and make more efficient procedures that are performed over and over. A macro is another form of saving a model. A model that is saved as a macro differs from models saved in the usual way in that a macro can be parameterized. To take a simple example: a ball that has been saved as a normal model-file will be shown as exactly the same size the next time it is called up. A macro that produces a ball can however be given a parameter value that tells the macro how big a radius the ball should have each time.

The macro window performs the following tasks:

- Shows macros: Plays saved macros
- Records macros: Saves all the activities of the user (Button REC)
- Executes macros (EXE button)
- Stops the execution of a macro (STOP button)

For further information about macros and the silux Script, please see the : *silux script Language Guide*

34 Command Window

Abb. 29: Command Window

You enter commands in the syntax of the silux script language in the Command Window. The command that you enter will be carried out as soon as you hit the return key, or will at least be recognized if the command is not meant to be carried out yet. You can also use the Command Window for entering commands during a simulation. During a simulation you could, for example, enter the command that the simulation should be stopped after running for 10 seconds:

```
STOP SIMULATION (TIME > 10.0e6) -- microseconds
```

For further information about the silux Script, please see the *silux Script Language Guide*

35 silux Script

A programming language was developed especially for the silux program. The use of this language is optional. The program will work without your ever using it, but the silux Script can make your work a lot easier. silux Script is an easy language to learn. Even if you have never used another programming language you will be able to learn silux Script just by using the silux program. silux Script can be compared with the EXCEL-macro or other similar languages in terms of the basic idea and the formalism. The vocabulary of silux Script is however entirely different and was produced especially for the needs of simulation technology. silux Script is a universal language that is used for many tasks in the program environment:

- Command language
- Recording
- Programming language
- Macro language
- Controls
- Program operation
- System optimization

Command Language

A user command will be carried out immediately or will become immediately active. If the command contains a condition, this condition will be taken into account. Commands are typed into the command window and executed by hitting the Return key.

Recording

When recording is turned on in the Macro-menu or in the Macro Window, all the actions carried out by the user will be automatically recorded in the Macro Window in the silux Script Language. The very same actions will be repeated if the user lets the recording play by clicking on the Execute button.

Programming Language

You can write a silux Script program either directly in the silux program or externally in a word-processing program and then transfer it into the silux program for execution.

Macro Language

Commonly used procedures or drawings can be saved as macros. Macros can be parameterized. Macros can be called up in the command window. Here, they can be given parameter values. Macros can also be assigned buttons in a so-called macro-table and then be called up simply by clicking on the corresponding button.

Controls

You can create simple control systems with the Create Control System dialog box. Generally however, you can also program control systems, especially complex ones, as silux Scripts. You can use them in the intended simulation and have them carry out their assigned tasks.

Program operation

You can pre-program the chronological and conditional course of the simulation. You can have the simulation stop if certain conditions are met. You can have the simulation start up again after certain, likewise pre-programmed changes, are made.

System Optimization

You can associate program management with an optimization objective. You can program an optimization strategy with silux script. In this way, it's possible for silux to repeat simulations over and over and, in doing so, to automatically optimize certain variables, parts, objects or movement procedures.

silux script programs can be saved onto external forms of data storage with the command *Save* silux script and then later re-loaded with the command *Open Macro*. The text will be opened in the so-called Macro Window.

For further information about silux script, please see the *silux script Language guide*.

36 Graphic Window

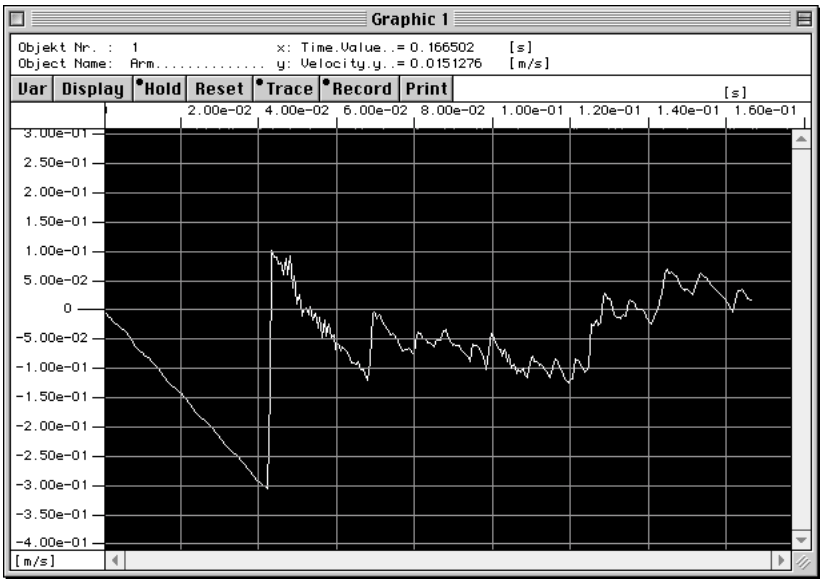


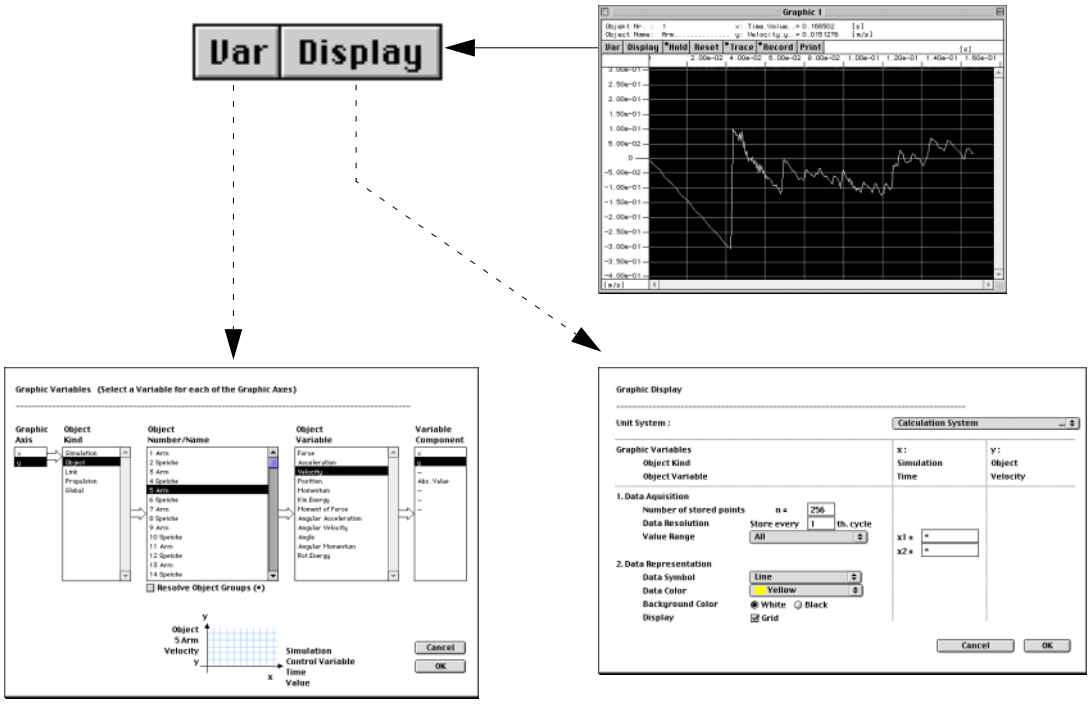
Abb. 30: Graphic Window

If you want to observe a specific physical variable of a certain object during a simulation, you can have this variable presented in the form of a graph. You could, for example, represent the value of the variable over time graphically.

For such a representation you would call up a graphic window from the appropriate menu function beforehand. The choice of the variables and the type of representation takes place in the graphic window itself. Various dialog boxes can be opened for these settings through a button bar in the Graphic Window.

One dialog box must be opened: that corresponding to the **Var** (variable) button. You need to tell the system which physical variables or events you want to see.

The button **Display** is facultative. They allow you to determine the type of representation and the scale. You will probably be satisfied with the standard settings in most cases though.



You can keep a great number of graphs open (on one or on several screens) at the same time. All settings can be individually defined for each graph. New graphs can also be opened during a simulation and already existing ones can be closed without disturbing the simulation process. Simulations and graphs management are processes that can run in parallel.

In the button bar of each Graphic Window there are also buttons that allow you to freeze the current event (Hold), reset it (Reset), or trace over the contents without erasing them (Trace). The numerical value of the most recently drawn point is shown numerically at the top of each graph.

37 Graphic Possibilities

The graphic tool used in the silux program can be used universally for any sort of data sketching. You can not only create graphics that display the chronological progression of a variable as a continuous curve along the horizontal axis according to a known pattern, but you can also draw curves that turn in place or that run in circles.

You can truly have any variable drawn with regard to any other. You are completely independent in your choice of which variables to place along the two graphic axes. The variable that you use along each axis can be chosen separately from the set of all possible variables. Theoretically, there is an almost endless list of combination possibilities.

Graphs that let you see the spatial track of a specific point of an object chronologically are also quite interesting. These graphs are produced in the following way:

- plot the x-component of the position of the object along the x-axis
- plot the y-component of the position of the object along the y-axis

Using these settings you can save the path of an object over the course of time graphically.

If you simply choose the position of an object, then the position of the reference point of the object will be drawn. If you only want to draw the trail of a certain point of an object, the following possibilities are available:

- Fasten a very small object (a sensor) to the point on the object that you want to follow. Now you can have the position or the reference point of this sensor drawn, like in previous examples. You can make the sensor so small that it does not affect the weight of the object at all..

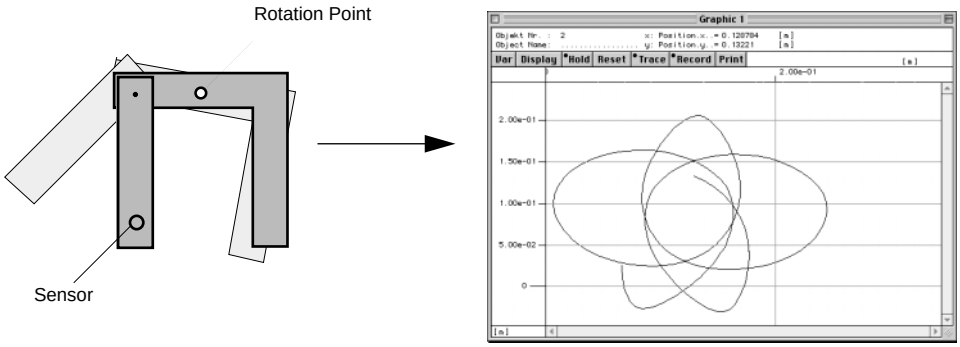


Abb. 31: The trail of a specific point of an object

In addition to all the variables of an object, you can also graphically represent the variables of the following types of objects:

- Links
- Propulsions
- Actuators
- Dampers
- Global variables (like total energy, total impulse, etc.)
- Group values of object groups (average, sum, etc.)
- Events (Switching of gears, states, collisions, etc.)
-

Note: Please see the chapter *Sensors* for further information about the sensors

38 Recording / Overview

Computing-intensive calculations are often carried out overnight or in the absence of the user. What exactly happened can be studied later.

With silux you can have simulations drawn while the calculations are taking place. We are speaking generally about picture and data representations. There are different reasons for wanting the simulation to take place while the calculations are going on.

1. You may want to see the numerical values of certain physical parameters. You may not want to only see the chronological progression of certain physical parameters (graphic representation), you may also want to have the numerical values of these variables explicitly drawn, for later use in another program (data representation).

2. Especially in the case of calculation-intensive simulations that, in extreme cases, may last up to several hours, you may want to have a sort of film recorded so that you will be able to watch the simulation later, in "real time". In this way you can gain an insight into the dynamic behavior of the entire system that might otherwise be lost in the case of a simulation that lasts a long time.

In the following, we differentiate between the different methods of representation that are available through silux:

- Graphic representation (data representation in graphic and numerical form)
- silux recording (geometric recording of the model)
- Quicktime (a graphic recording of the model)
- Data recording (numerical values of physical variables, measurement values)

All three types of representation can take place simultaneously during a simulation. All the relevant rules for each form of representation can be given separately.

39 silux Recording

The system does not save the picture-information during the recording of a silux simulation, but rather the geometric position of all objects. Because of this, when you replay the simulation you can look at the individual pictures of the recording in any scale and at any section, exactly as the scale and roll-bar settings allow you to in the real model. This is the most fundamental difference in regard to the Quicktime recording method. silux Recording allows you to zoom in to the geometric details and look more carefully at the particulars. Even objects that are located beyond the current screen section are saved. Later, when looking at the recording, you can move these areas into the visible portion of the screen with the roll-bar.

silux Recording is then the suitable recording method for analytical purposes.

You can determine the frequency of the recording in a dialog box. You determine this frequency according to your own idea of what simulation detail and resolution are sufficient for it to be comprehensible later when you're watching it. A typical period lies between 10 and 100 calculating-cycles.

(Note: The recording will possibly be expanded later such that physical variables could also be saved at the same time. Then you could also read off, for example, the speed of the individual objects from every picture.)

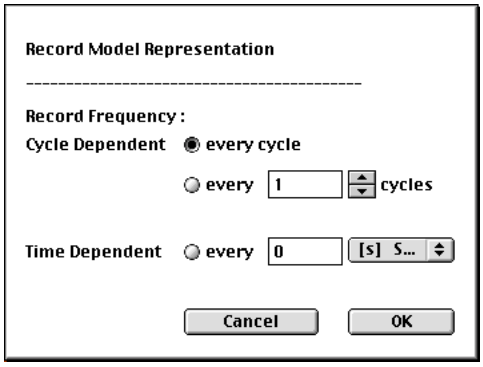


Abb. 32: You determine the frequency with which the state of a running simulation is recorded.

40 Replaying silux Recording

Abb. 33: The keyboard for Replay-management

A recording can be opened with the menu-command **Replay** in the file menu. The so-called replay tool is presented whenever a recording is made. This tool allows the user to control the playing process, just like with a video-recorder. The **Setting** button allows you to open a dialog box in which you can change the play-frequency of the recording. In this way, you could specify, for example, that the system should draw only the X'th saved record on the screen so that a certain play-speed can be upheld. How fast the recording is re-played can be specified in the Replay Setting dialog box.

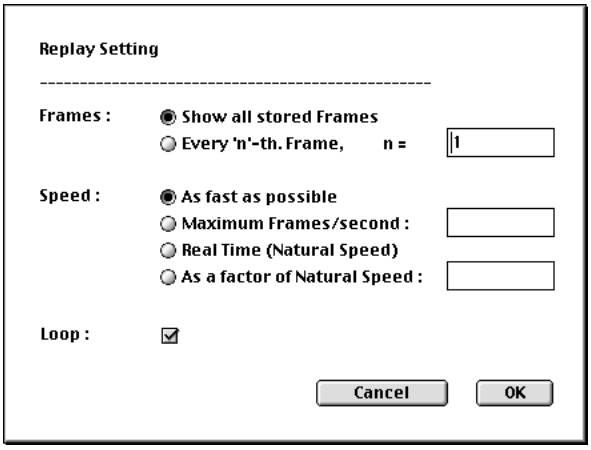


Abb. 34: Dialog Replay Setting

41 Quicktime

Quicktime is digital video-technology that was developed by Apple Computers. This technology is available for both Apple systems and for MS Windows-based operating systems. Video sequences that are based on this technology can, without being changed, be both shown and associated with applications on Macintosh and on Windows Systems.

Quicktime is a standard that also provides everything associated with the digitalization of films. A fundamental part of this is data compression with many different compression-algorithms and methods and sound-tracks. Quicktime allows you to make "feature films" in the true sense, but you can also digitize and compress "freeze-frames", photos, technical diagrams, animation, movies, and so on.

The basic elements in Quicktime are pictures that are compressed either individually or as a group. No information about the form or position of individual objects in the pictures is saved in Quicktime. Quicktime recognizes only the pixels or the numerical representation of these pixels. These figures are what is compressed by different algorithms during compression.

If you activate the Quicktime function in the silux program, then you should determine the frequency with which new pictures should be added to the video you want to create during recording. This works exactly the same way as in silux recording. The size of the pictures in the video will correspond to the current size of the model-window on the screen. The Quicktime function will save copies of the model window in the video automatically, according to the programmed rhythm. You determine exactly what should be saved in the video by setting the desired section of the picture on the screen. What you see on the screen will be recorded exactly as you see it.

No physical values or information can be taken directly from a recording that has been made with Quicktime. A Quicktime recording is a video and nothing else. But, as far as presentations of the movements of a model go, Quicktime videos are the nicest to look at and the provide the smoothest replays. Quicktime is then the preferred recording method for presentations. For playing Quicktime movies you can use standard playing programs (Simple Player, MoviePlayer, etc.) that are delivered as system components with the operating system.

Individual pictures are compressed "ad hoc" during the recording and the adding of pictures to Quicktime movies. This prevents an early overloading of memory capacity and substantially increases the amount of data that can be stored. The pictures are compressed individually and not as a series. This means that Quicktime movies, as created in silux, have not yet been optimally compressed. You can, however, take a copy of the existing film and compress it further with a suitable editing program. The following programs can be used for this task:

- MovieShop © Copyright Apple Computer Inc.
- Adobe Premiere™ © Adobe Systems Inc.

Generally speaking, Quicktime movies that are created with the silux program can be compressed to a relatively high degree, higher than video sequences that have been recorded with a video camera. This is because the ”picture information”, as compared to that of normal video film, is much more simple and uniform.

42 Drawing

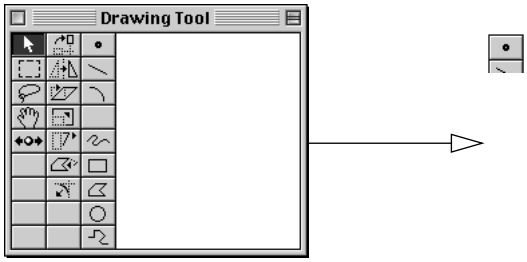


Abb. 35: Drawing Functions

The drawing functions of the Drawing Toolbox work in the silux program exactly like they do in all other drawing programs. If you can already use another drawing program you shouldn't have any problems using these functions. Just use these buttons intuitively.

For more exact information about the drawing functions, please see the *User Manual*

43 Analytical Geometry

In silux we mean the following with the term ”analytical geometry”:

- In the silux program you can not only select entire geometric objects just as you can in all drawing programs, but you can also select individual points of an element.
- These points can be used then for the drawing process of the next element.

To select a point, move the cursor along the contour of the appropriate element. Generally, you will then be shown a white or (hollow) arrow: this indicates that the system has acknowledged the contour. If you click on the mouse button at this point, the whole element will be selected. If you do not click on the mouse-button, but rather move further around the contour, you'll see that the cursor turns into a small circle at some points. These are the points that you can select within the corresponding element. If you do select such a point, a single selection-mark (a small black square) will appear where the point is to show you that the selection has been made. You can select additional points in exactly the same way as usual with the Shift button. You can select a maximum of three points. If you try to select a fourth point, the first point will be erased, and so on. You can de-select or erase these points the same way as you do it with elements: simply click near the point, but in a blank space. Point selection generally works only with raw data and is not possible with physical objects.

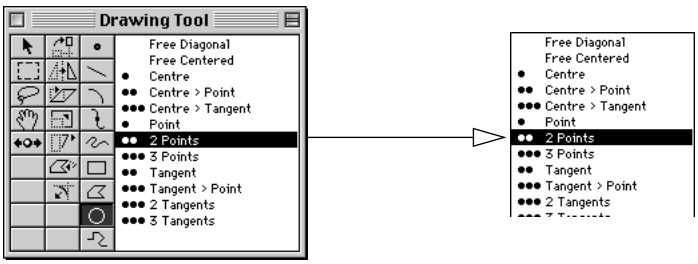


Abb. 36: Drawing Methods (a circle through 2 points has been chosen).

The selected points will be used further in the drawing function. Each drawing method indicates how many points should be selected before the function can be used (•••). Corner points, as well as end points, middle points and points of intersection can be used as points.

44 Modifying Elements

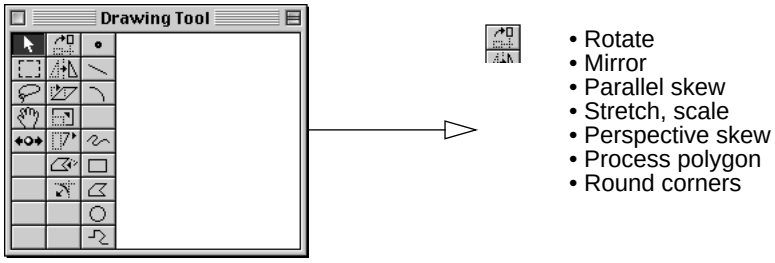


Abb. 37: Modification Functions

In exactly the same way you can use different methods to draw a certain element, you can use sometimes many different methods for the modification of these elements. The methods available for element modification are presented as a "choice list" that you can click on to choose the desired method in the far right of the Drawing Tools box.

Additional operations for the elements:

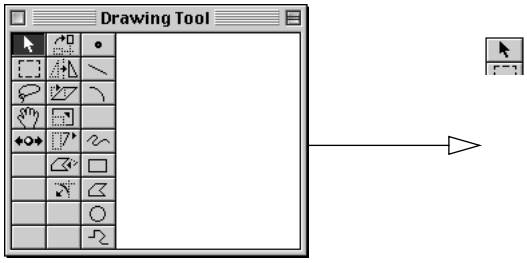


Abb. 38: Additional Functions

The first column of the Drawing Tools box has to do with the selection of groups and with moving the drawing as a whole. These operations function in exactly the same way as in all programs that use this function. For further information about these functions, please see the *User Manual*.

The arrow symbol, in the upper-left corner, is the default symbol: it is always activated when another operation is closed. You can also select this arrow symbol when you want to de-activate a previously chosen button and the corresponding function. Elements and objects are selected and buttons and roll-bars are used/activated with the arrow symbol in the model window.

45 Data Import and Data Exchange

The silux program is not a CAD-program but is in particular, among other things, a complementary software to existing CAD-systems. silux do not intend to compete with existing CAD-programs, but rather aim to provide a powerful complement to the existing infrastructure in the engineering field (CAE). For this reason, it's important that silux can exchange data with as many CAD-programs as possible.

The silux program works with the DXF data exchange format. In this way you can transfer geometric data from other (imported) drawing or CAD-programs and then later move them back. The geometric data loaded into the silux program has a "raw data" status at first. Raw data can be completed with physical properties and in this way raw data is developed into complete physical objects. In silux, such complete data can take part in all simulations because they are now complete physical objects. Intelligent functions and individually-set standards help you with the transformation of raw data to objects.

silux has its own exchange format (TEXT) that covers both the geometric data and the physical data of a model. Files that are saved in this format can be loaded by word-processing programs. TEXT files can similarly be edited or changed with a word-processing program. This data-format is however, probably more important in the case of a comprehensive platform data exchange. The TEXT exchange format is, in regards to changes from version to version, ability to operate on different computer platforms, etc., definitely the least sensitive exchange format.

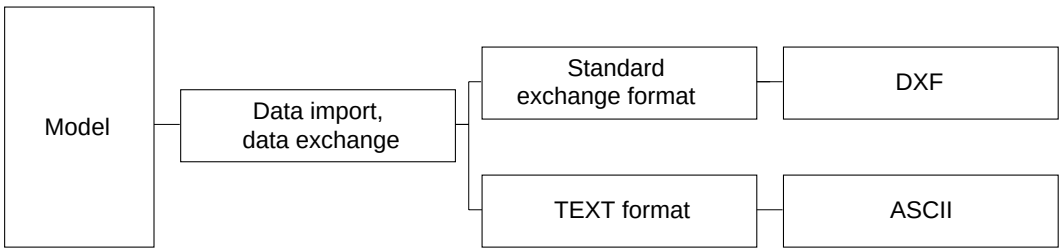


Abb. 39: Data exchange choices

46 Object = Geometry + Physics

The things that you draw on the screen are elements. Elements are raw data without any physical (characteristics). They are simply geometric lines, curves, figures, etc. To bring this raw data ”to life” or to give these elements the ability to take part in a physical simulation , they must be expanded with some physical information. For example, a physical object has to have a mass, a moment of inertia, etc. The transformation of raw data elements into physical objects takes place in a dialog box especially created for this purpose.

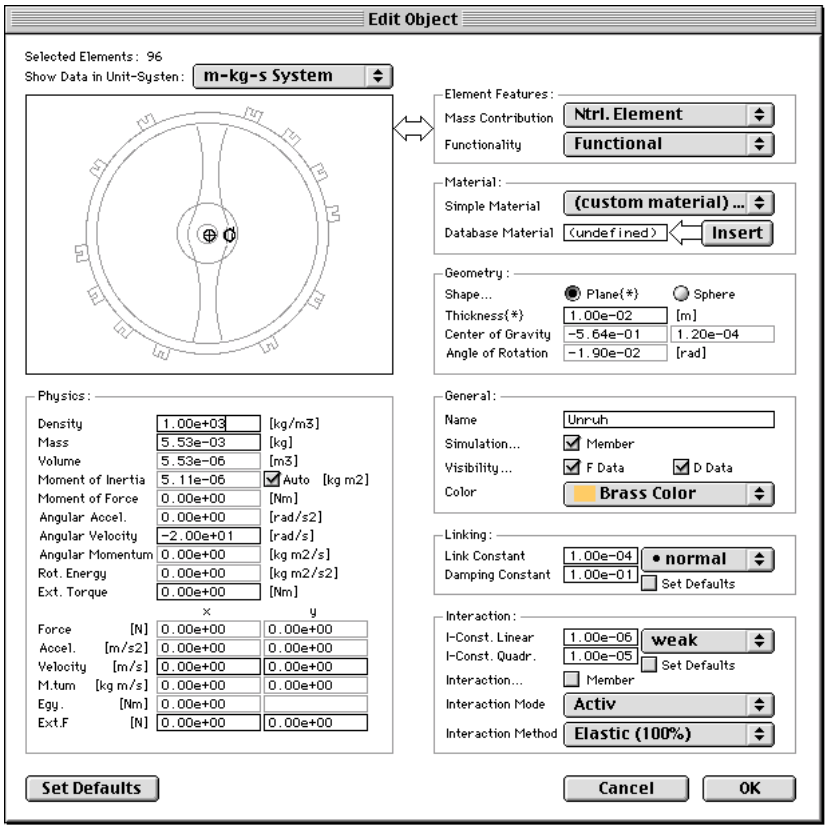


Abb. 40: Create/Edit Object Dialog box

You do not need to understand too much physics because the program does most of the work for you. You don’t have to know, for example, what a moment of inertia is or how you calculate it. The program does this work for you automatically. There is no reason for you to

be afraid of this dialog box just because it's made up of a lot of fields and buttons. A lot of the information involved is facultative and doesn't necessarily need to be filled in or selected. The contents of many fields can be derived from other fields. In extreme cases you may not even need to do anything, just click on the "OK" button immediately after the dialog box opens. This will create an object that will "survive" though possibly with characteristics other than what you'd imagined. With reference to biology, you could look at silux's physical objects as single-cell creatures. They are the smallest creatures capable of life. In silux, a single object contains all the necessary information to be able to "do" physics, to determine the behavior of the object in the group and in relation to other objects, to save the type of representation on the screen, and so on. Objects can be connected to other groups later. These objects can later be defined by other group characteristics, but the core of its behavior is contained in the object, in the physical building-block of the silux program.

Specific objects can also be created by macros, just because macros have been developed for this task. Some macros intended for such purposes have been provided with the program, but you can create others for the purpose of specific fields. In other words the detailed filling-in of the dialog box shown above, as in other dialog boxes, can be exactly recorded or manually programmed in a macro.

We can summarize the flow of data in silux, as described so far, in the following way:

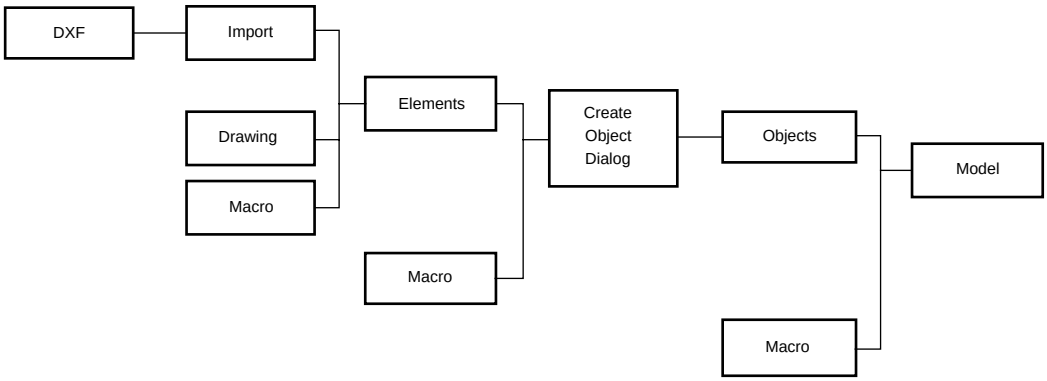


Abb. 41: Data flow in the program: from raw data to a model

47 Creating Object Groups

The silux program allows you combine individual objects into a group and then, as a result, they will appear as a group. The individual objects move to the background and the properties of the entire group become most important in terms of appearance. So-called ”group characteristics” can be defined. ”Group characteristics” are then valid for all objects that belong to a certain group. If an object of the group is clicked on, then the entire object-group is selected. Functions that follow up a selection, such as erasing, duplicating, or other functions consequently affect the entire group.

These group characteristics are normally defined by the program itself (e.g.: cables)

48 Moving Objects and Object Groups

Moving raw data and moving objects is not exactly the same. Raw data can, in principle, be moved in any way without creating any problems. Objects, however, are physical bodies. In order to be, and to remain, physical bodies, they can move in a pre-defined physical environment and do anything that does not lead to contradictions within the model. As long as a simulation is in progress, the program makes sure that physical boundaries are not crossed. If you, the user, however intervene manually in the model, you may do things that you're not really supposed to do. In this case, it might be that, at the same time these "non-permitted" changes are being made, other steps must be taken in order to ensure the integrity of the model. The contradictions that might arise are caused less by profound physical laws, but rather much more by direct geometric problems that arise when the objects are moved.

The program tries to solve these problems in two ways:

- The program tries automatically, whenever possible, to take steps that correct for the action you've taken.
- If this is not possible, for whatever reason, you will be sent a message. The message is meant to communicate the contradiction. Under certain circumstances the program will not carry out the action that you request.

Example 1:

- The user moves an object that is connected to many other objects by a long chain-link.

The program drags the entire chain of objects when the first object is moved: the program takes advantage of the ability of each object to move a bit. If, somewhere in the chain, an object has to move out of its own space, for example, if a wheel or lever is forced to move from its axis, then the program will sound an alarm.

Example 2:

- The user tries to move an object that is linked to a fixed-point from its axis.

The program will show an error message and will not carry out this command.

Example 3:

- The user moves an object onto another object so that the two objects cover each other.

In this case, the program will check to see whether or not this is allowed: the user may or may not have specified that a collision is possible between the two objects. The program will offer you the possibility of introducing such a rule "ad hoc".

(Please see: Interaction Rules, Controlling Collisions, Types of Collisions, Collision-Management)

Note:

Object groups are always moved as a whole group. A cable is moved or turned translatorally as a whole when you either move the cable with the mouse or turn it with the turning function.

49 The Material and Semi-finished Product Database

The different dialog boxes that you'll want to use to define objects or object groups offer you access to databases containing pre-specified or semi-fabricated material..

Dialog	Type of data	Examples
Create Object	Objects	Metal, synthetic material, glass, wood, etc.
Create Spring	Springs	Tension spring, compression springs
Create Cable	Cables	Strand wire, traction rope, suspending wire, etc.

Tabelle 3: Materials and Semi-finished Products

You can access this data through the standard window Material Tool. Do the following in this window:

- Load a material or semi-finished product file.
- Choose a material or part in the tree-structure.
- Transfer the material or part with all of its properties to the program by clicking on the button.

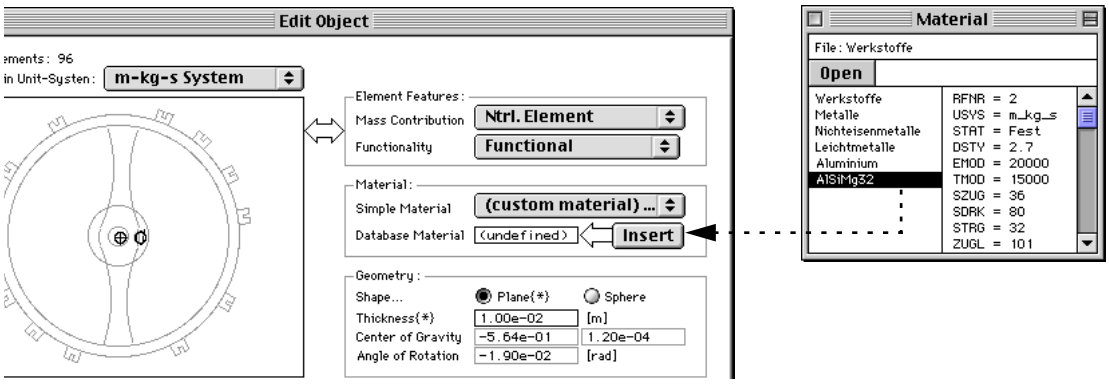


Abb. 42: Transferring Material from the Material Database

All material files have the same data-structure and can be brought to the screen with the same Material Tool throughout silux. The data-structure is described in the annex to the User Manual.

50 Cables

A cable is a chain of objects linked together. The length of the individual cable elements and the physical properties of the individual links between the cable elements determine the properties of the cable as a whole. You can create cables with an easy-to-use tool that supports the definition of the cable properties and the creation of the cable as a whole.

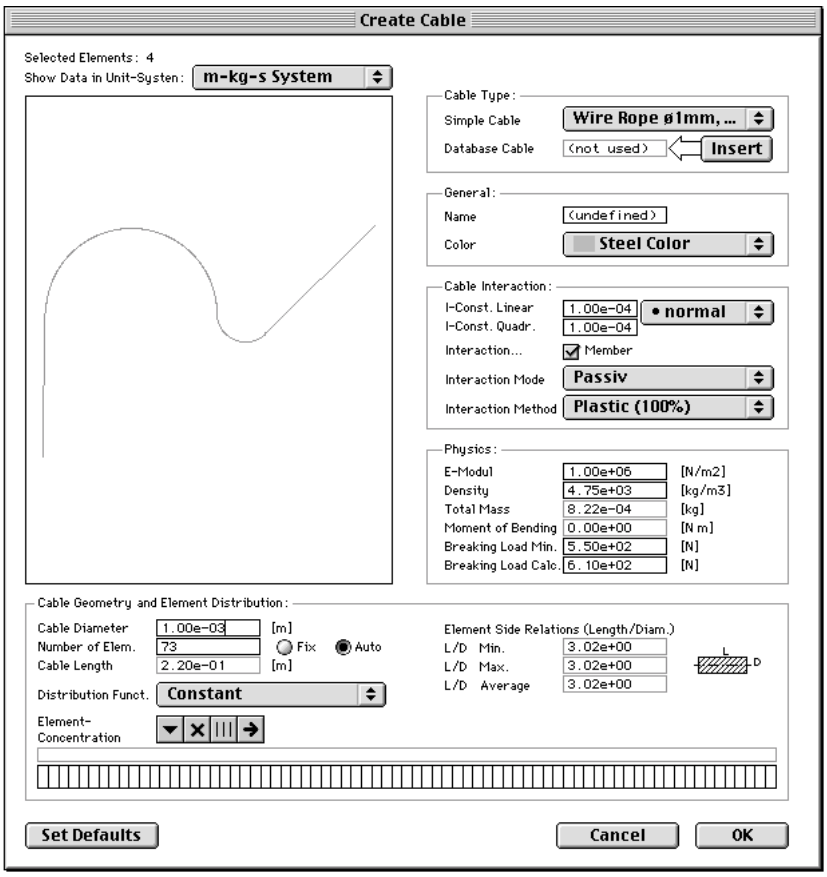


Abb. 43: Create Cable dialog box

The creation of a cable can be summarized as involving the following actions:

Draw an opened or closed curve-path. A special drawing function will help you to lay out the path, which consists of lines and curves. The same drawing function will help you to lay

cables around cable-wheels. This function can be activated in the Drawing Tools dialog box.

- Select the cable-path and open the Cable dialog box.
- Define the cable properties. Generally, you'll choose a cable from the database. The diameter of the cable and material properties of the cable are then automatically set. You can also specify the number and the distribution of the objects along the cable.
- The cable, with all of its objects, is automatically created when you leave the dialog box. The links between the objects are individually measured such that the cable, despite any possible differences in cable-element lengths, shows constant properties along its length..

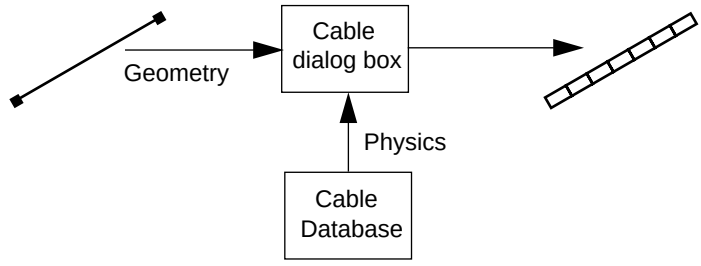


Abb. 44: Schematic cable creation

51 Helical Springs

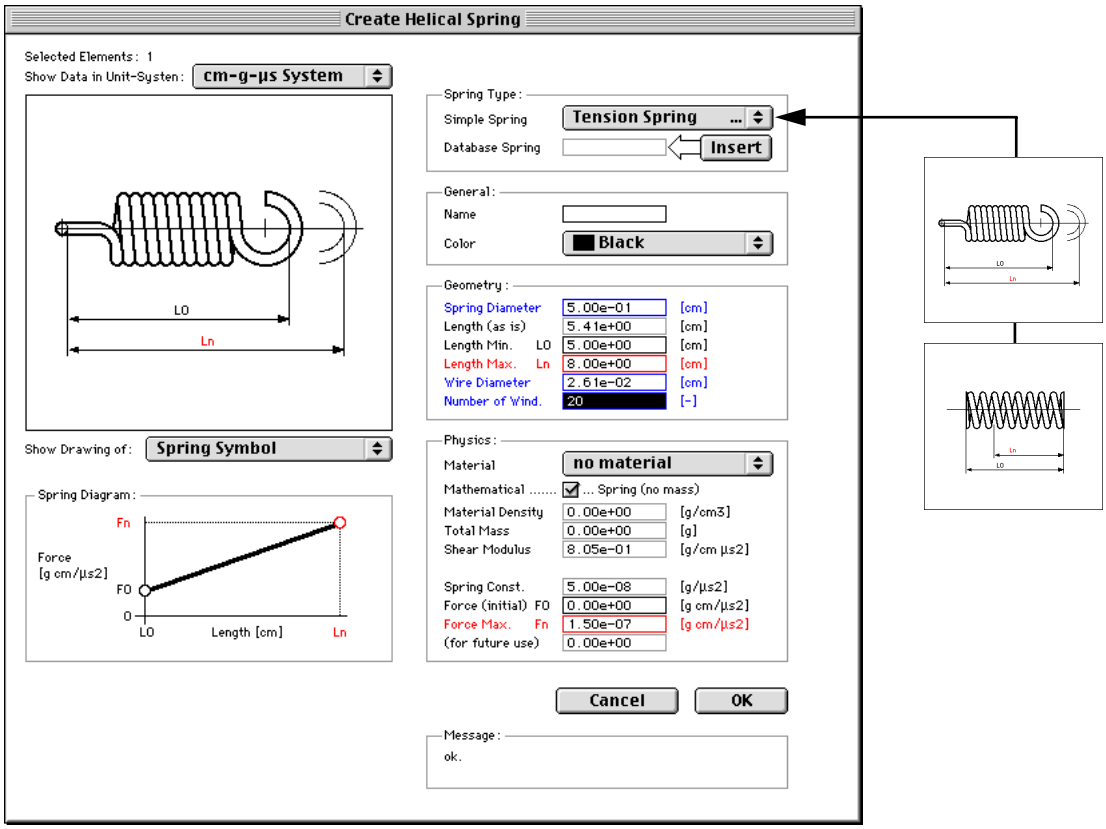


Abb. 45: Create helical spring dialog box

Before you call up the spring dialog box, draw a line in the model. The line should represent the position, orientation and current end-point of the spring. With this information, the system can fit the spring correctly into the model.

- Select the line that you've drawn and open the Spring Dialog box.
- Define the spring characteristics. generally, you'll select a spring from the data-base. The spring geometry and all material properties of the spring are then automatically set.
- The number of coils, the diameter of the spring, the strength of the spring, etc. can however also be changed. The system will immediately calculate all the dependent variables after every change you make. If the permissible bounds of a variable are over-

stepped or if a spring cannot be calculated from the values that have been entered, a message and the corresponding cause will be shown. The system tries to keep the spring dimensions within the variable boundaries of a spring that can really be created.

- The spring will be created automatically when you leave the dialog box and will be placed where you have drawn the initial line

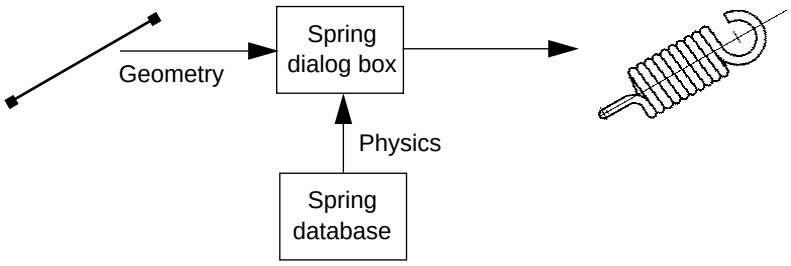


Abb. 46: Schematic Spring Creation

52 Gears

Gear wheels can be created from a basic object element. You need to imagine the creation of a gear wheel exactly as it works in reality. The gear wheel consists of more than just teeth. A gear wheel has a bore-hole, a hub, possibly spokes, etc. The teeth aside, a gear can be a relatively complex component. As long as it doesn't have any teeth, the gear wheel is, in the silux program, just an object like any other. The Create Object dialog box provides all the tools we need to create this object. The teeth will be added on to the otherwise completed raw object at the end, just as they are in reality.

The creation of a gear wheel works as shown in the following scheme:

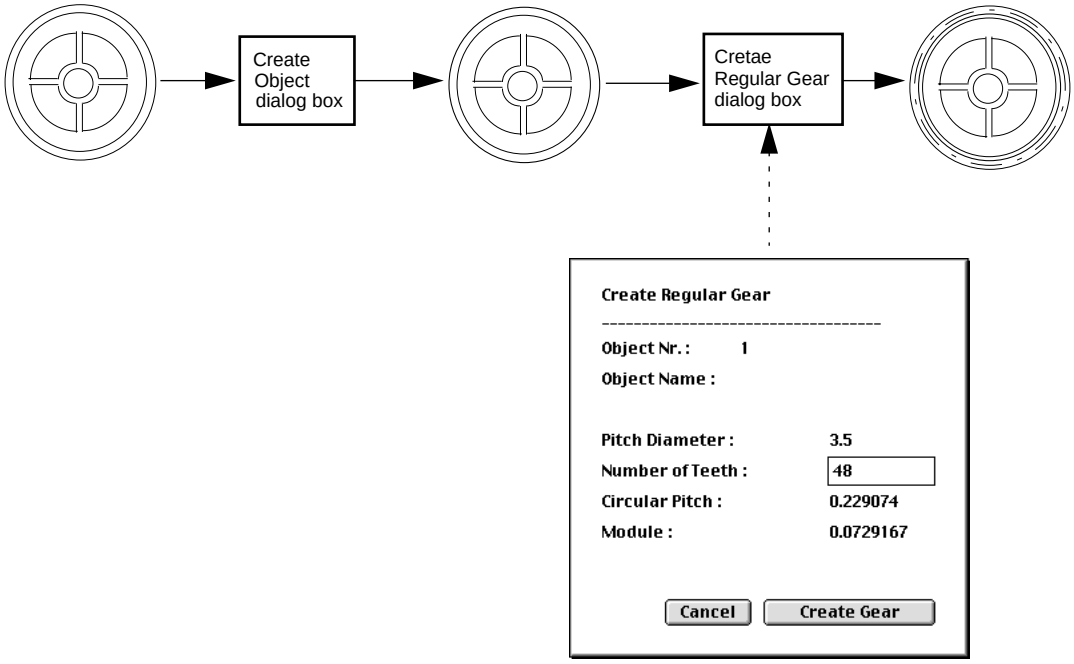


Abb. 47: Schematic Gear Creation

The system will interpret the largest functional circle of the object as the pitch circle in the *Create Regular Gear* dialog box. The three, inter-related variables, number of teeth, circular pitch and module can be changed interactively in the dialog box.

53 Linking Gear Wheels

The linking of gear wheels happens automatically. You don't have to worry about whether or not gears that are meant to work together will actually run or drive each other smoothly. Each time you create a new gear wheel, the system will check to see which gear wheels are in operation and will check the elementary requirements:

- Are both objects gears?
- Do the teeth have the same module?
- Is the distance between the gears permissible?

An error message will appear in the case of gear wheels whose outside diameters intersect each other but whose part boundary circles do not contact each other.

54 Dampers

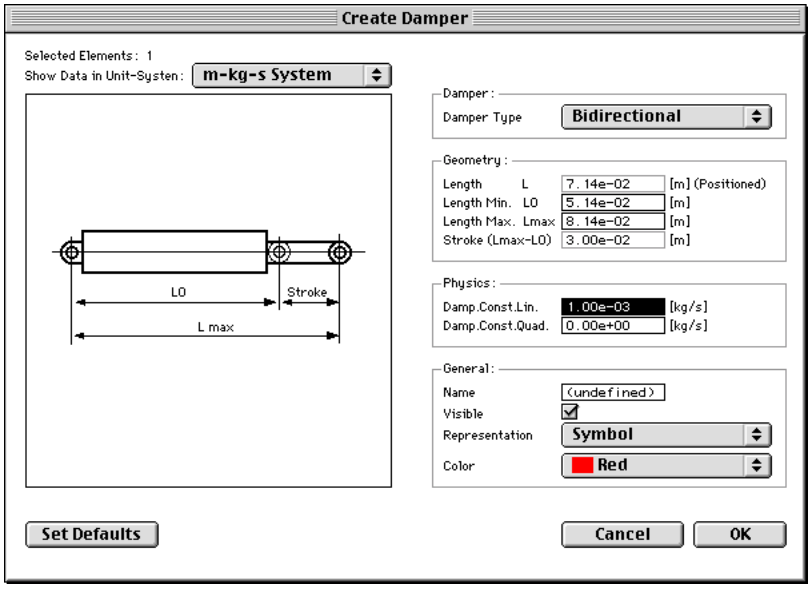


Abb. 48: Create Damper dialog box

A damper damps the relative movements between two objects. One of the two objects may also be a fixpoint. A damper creates a restraining force as a function of the relative radial velocity between the two objects.

Before you call up the Damper dialog box, draw a line in the model. The line should represent the position, orientation and the current end points of the damper. The system can then fit the damper into the model correctly with this information.

- Select the line that you've just drawn and open the Damper dialog box.
- Define damper properties and specify other information.
- The damper will be created automatically when you close the dialog box.

The damper will still be loose, that is to say not yet connected to the two objects whose relative movement it is meant to dampen, after its creation. It can still be moved and turned. for information on linking dampers to objects and fixed-points, please see the *User Guide* .

55 Actuators

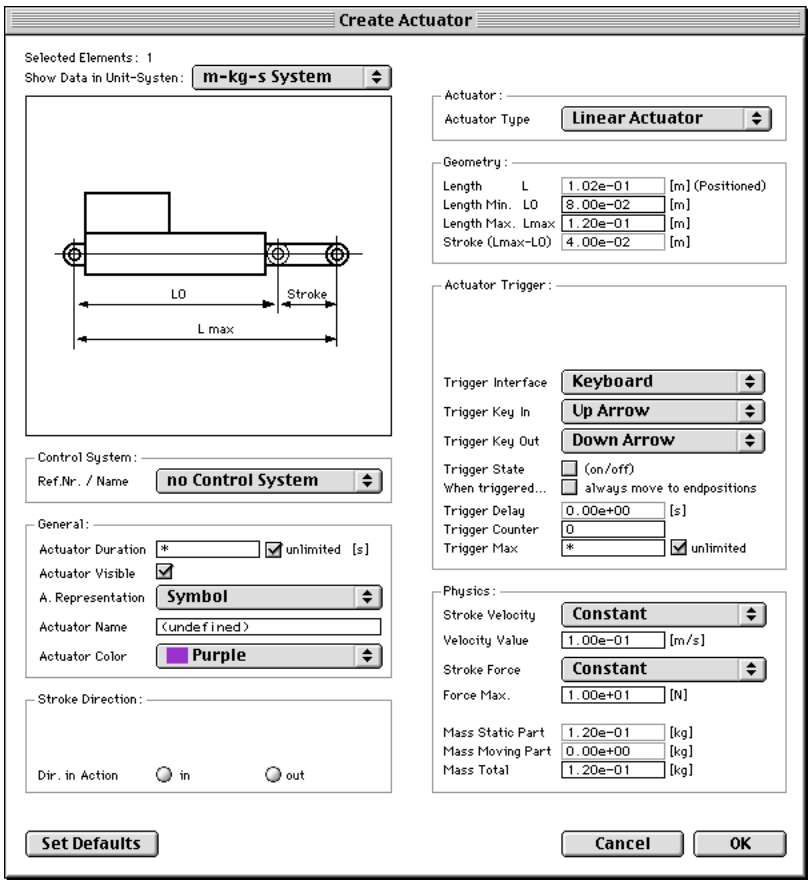


Abb. 49: Create Actuator Dialog box

An actuator is a linear correcting element or a motor. An actuator will change its length or angle of rotation (stepping motor) based on a given signal. An actuator can be controlled either manually with an interface (keyboard) or automatically with a control system.

Before you call up the Actuator dialog box, draw a line in the model. This line should represent the position, orientation and the current end points of the actuator. With this information, the system will be able to fit the actuator into the model correctly.

- Select the line that you've just drawn.

- Define actuator properties.
- The actuator will be created on the selected spot automatically after you leave the dialog box.

The actuator is still loose immediately after it has been created and not yet connected to the two objects whose relative distance it's meant to control. The actuator can be moved and turned. For information about linking an actuator to objects and fixed-points, please see the *User Manual*.

56 Propulsion

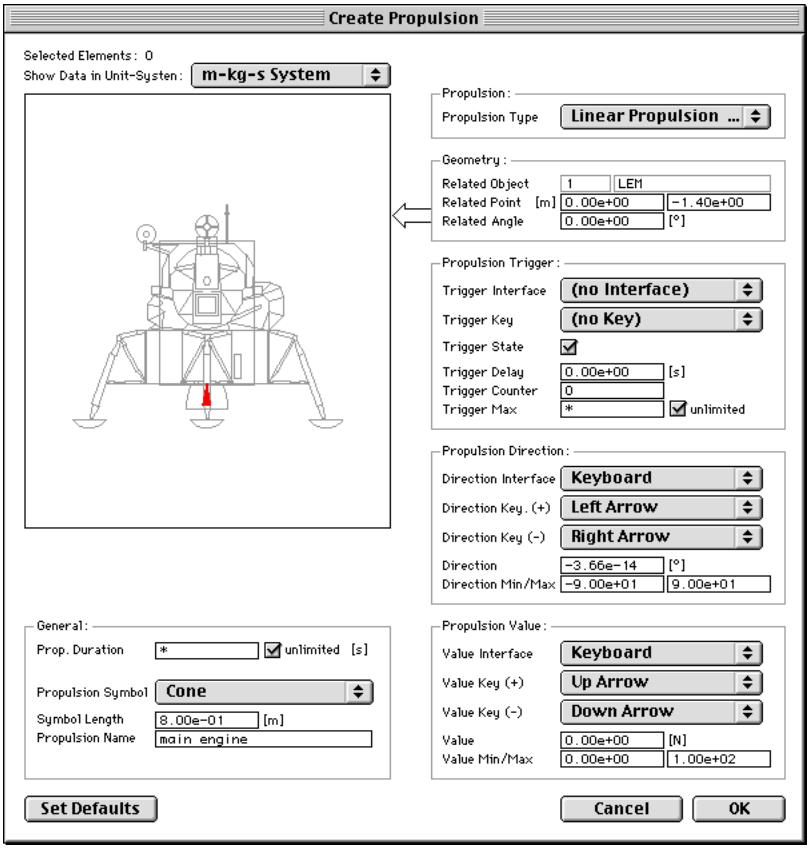


Abb. 50: Create Propulsion dialog box

Propulsion is a force-function that exerts a force on a specific object. Propulsion is therefore linked to an object. If you want to create new propulsion, you'll need to select an object before you call up the Create Propulsion dialog box.

The dialog box is opened with the menu-command Create Propulsion or, when the propulsion already exists, is called up from the Tool Window.

You can define the properties of propulsion yourself to a large extent. For the interactive placement of the propulsion with the "mother object", both elements are displayed graphically. For further details, please see the *User Guide*.

57 Links

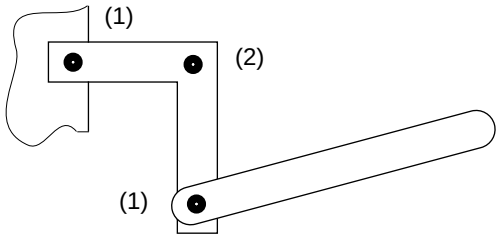


Abb. 51: A lever associated with links

A link is a dot-shaped connection between:

- Two objects (1)
- An object and a fixpoint (2)

You can imagine a link as representing a needle-shaped axis that sticks through both objects. If one of the two objects is a fixed-point, then the link is absolutely fixed. The object can then only turn around this point. If both linked elements are objects, then they can only move together under the specified conditions.

A link (object to object) is created in the following way:

- Draw a point (with the Drawing Function) where you want the link to be.
- Select this point and the two objects*
- Choose the menu-function *Create Link Normal*

A link (object to fixed-point) is created in the following way:

- Select an object and a fixed point*
- Choose the menu-function *Create Link Normal*

Objects that are already ”naturally” connected to each other or that can be turned together do not need to be additionally joined by a link.

There’s an alarm function in the program: this will stop the simulation if the maximum force allowed in a link is being surpassed.

(*) You can select several objects at the same time by selecting the first object as normal, with a mouse click but then selecting all further objects with a pressed-down Shift key.

58 Edit Link

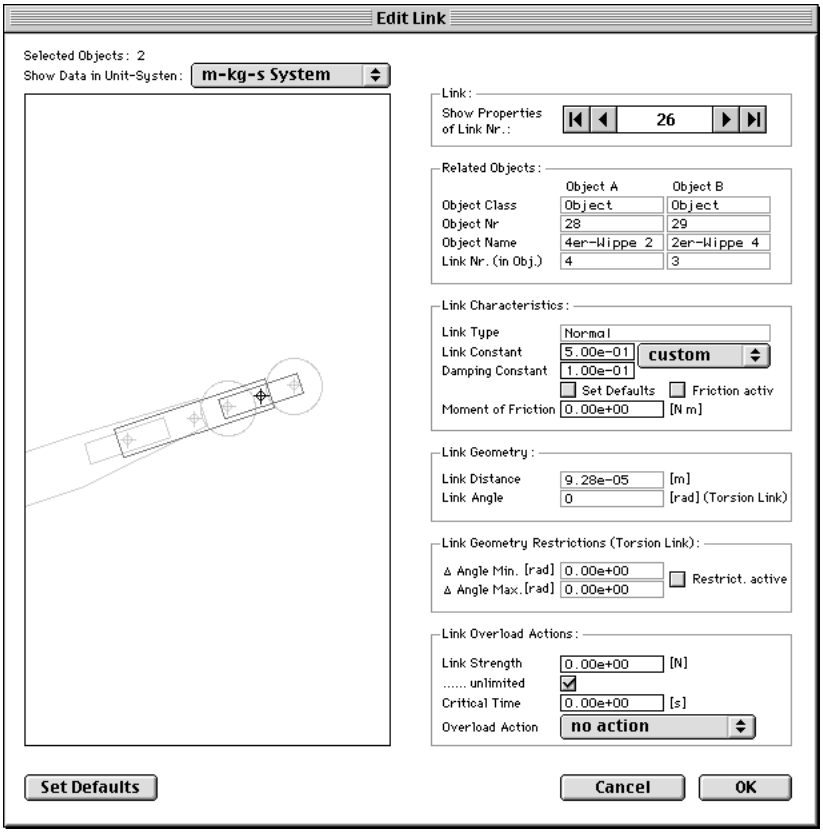


Abb. 52: Edit Link dialog box

The properties of a link can be specified or changed only after the link has been created. During the creation process, so-called pre-settings are used. If you want to change the standard settings of a link, then you should choose the corresponding link in the List Tool by double-clicking on it. The dialog box shown above will then open automatically and the link will be shown both in respect to its geometric position and with its properties.

For information about the specific effects of each link variable, please see the *User Manual*.

If you want to edit or just look at additional links, you don't have to leave the dialog box that you're in. You can use the four arrow buttons in the upper part of the dialog box to "scroll" to

other links. The graphic position-indicator in the left-half of the dialog box will make it easier for you to identify the individual links.

59 Gravity Fields

We differentiate primarily between two different types of gravity fields:

- Constant, uniform fields
- Central fields

Constant Unidirectional Fields

For tests on a laboratory scale we need a gravity field that works with the same force at every area and points in the same direction overall (generally vertically down.) We can activate such a gravity field in the menu. If we don't make any further specifications, then the field will have the force of the so-called standard-Earth gravity field. That's generally what the user expects.

If you want to choose another value or direction of the force however, then use the Edit Gravitation Field dialog box (it can be called up from the Menu or the Tool Window).

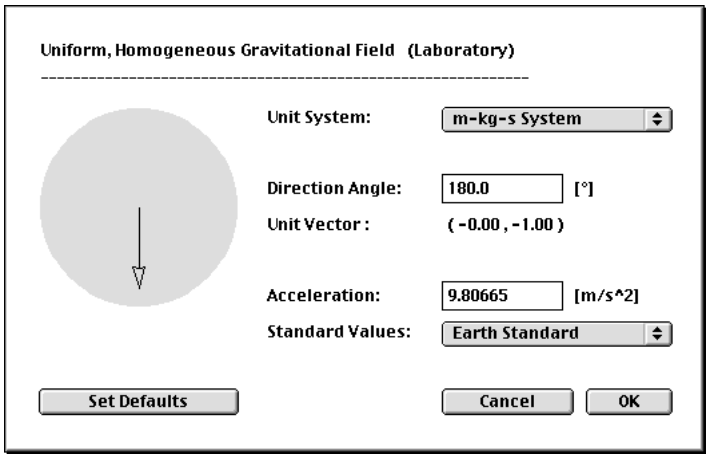


Abb. 53: Gravity Field (Laboratory) Dialog Box

You can carry out comparative experiments on the Moon, Mars, or on any other planetary body. You can set any value for the force field.

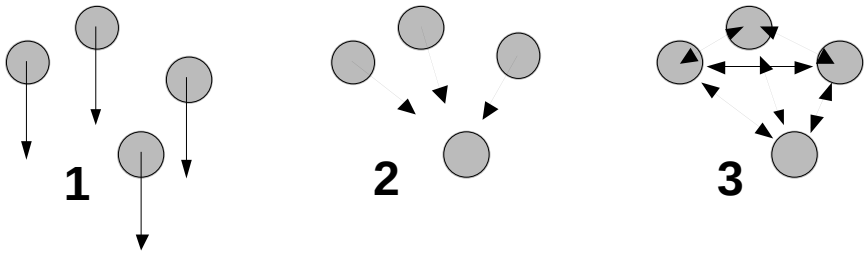


Abb. 54: Three possible implementations of gravity: 1. Constant, uni-directional fields; 2. A central field for individual objects, here for 1 object; 3. A central field for all objects.

Central Fields

Central fields can be used generally for all objects (case 3) or for individual, selected objects (case 2).

Case 2

If we want to observe several satellites in the Earth's orbit, we can just use the gravitational field of the Earth. All the satellites within this force field will orbit in their expected paths. The essential element here is the force of gravitation that the Earth exerts over all the other objects. It doesn't make any sense, in this case, to pay any attention to the individual gravity fields of the satellites (that is to say to the gravity fields that each of these objects exerts on the Earth) because it is so small that that it will only appear about after the 77th spot to the right of the decimal point! The situation is however different if a moon of a substantial mass becomes a moon in the true sense. The gravitational force of the Moon on the Earth cannot be ignored. If you wanted to simulate the flight path of a spaceship from the Earth to the Moon, you would activate the gravities of both the Earth and the Moon, but not that of the spacecraft.

Case 3

Generally speaking, you activate all gravity fields when you're simulating the interactions between many objects that are of the same scale or of different scales that cannot be ignored. Models that belong to this category are galaxies, the solar system, a planet and its moon or a collision between celestial bodies. For research into such collisions people often (break down) the individual celestial bodies into fragments that are then hold together only by their own mutual gravitational forces. In such cases, the gravitational forces of each fragment in the field of influence of each other fragment will be calculated.

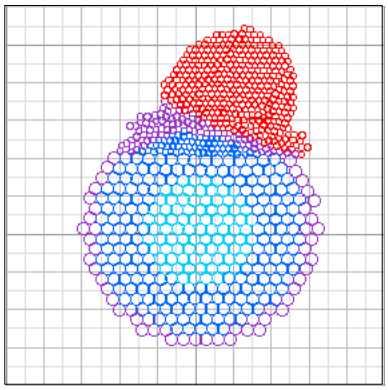


Abb. 55: The collision of a planet of the size of Mars with the Earth. (See: silux Samples, The creation of the Moon)

In any case, central gravity fields should only be used for astronomical calculations. Never use these fields for laboratory experiments. The constant, uni-directional field is always the correct field under lab conditions.

60 The Simulation Area

Most programs that you have run into until now have a specific document size. That's true for practically all text, drawing or CAD-programs. The individual sheets and their boundaries are generally visible on the screen. You write or draw on a virtual piece of paper. It is often true that you cannot write or draw outside the boundaries of this piece of paper. And this is surely rightly so... The ultimate goal of all these programs is to create a printed product. In the end, you want to have the product on paper.

In the silux program however, there are no such document sizes or document boundaries. A model is not restricted to moving within the limits of the size of a sheet of paper. A simulation really has nothing to do with a printed document. The goal of the whole process is, in the end, the simulation itself and this happens in the computer or on the screen. There are perhaps other limitations that will hinder movement in one direction or another, but there "sheets" have no edges. A model can take up time and space freely.

These circumstances mean that a model can be moved practically unlimited distances in a specific direction with the roll-bars. This is good because maybe we want to follow a model moving away from us over great distances. This special setting of the model window has no serious consequences in regards to the roll-bar. The only roll-bar function that doesn't operate as usual because of these special conditions is the position-markings of the roll-bar. The position markings are frozen to the middle setting in a model window and cannot be moved. The arrow buttons of the roll-bar and the sideways forward- and backward-scrolling work as usual however.

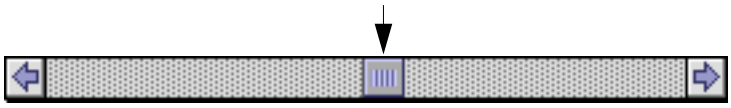


Abb. 56: The position markings of a roll-bar

61 Starting and Stopping Simulations

You can start the model on the screen at any time. In general, it's a good idea to make a copy of the current state of the model because the models are often changed as a whole through the simulation. You need to imagine the model as existing in time and space. The procedure cannot be undone so it is better to have a copy of the original state of the model.

- Use the menu function *Save as*... to save a copy of the model.
- Start the simulation with the menu-command *Run*.

Now you can sit back and watch what happens. It won't disturb the simulation if you make a detail larger by using the scroll and zoom functions. The simulation will keep running during these procedures. If you don't do anything, then the simulation will run on forever, until the end of time, or until the cleaning lady switches off the computer.

You can stop the simulation at any time and then start it up again later.

- Stop the simulation with the menu command *Stop*.
- Save it.
- .....
- Open it
- Re-start the simulation with the menu command *Run*.

62 Stopping and Changing Simulations

It is not true that the model cannot be changed or is otherwise ”fixed” once the simulation has been started. The model can, in principal, be stopped and modified as many times as you want. Each time you stop the simulation you can change every variable of every object. You can do this with the same freedom you had before you started the simulation. A simulation that has been stopped is, in regard to the changes, is as accessible as it was before the original start of the simulation.

- You can change parts of the model graphically and interactively on the screen by, for example, moving or rotating certain objects.
- You can also select individual objects and call up the Edit Object dialog box to, for example, change the velocity or some other variable of the object.
- After these modifications have been made, the simulation can be re-started from its new, modified situation.

You should treat the model as you would treat such a model if it really existed. You should be able to intervene in the model with your hands, you should be able to ”touch”, move and rotate the model but you should also be able to leave it on its own and just watch. You should look at your model and its environment as a laboratory.

63 Influencing the Simulation

In silux, you can intervene directly in a simulation and ”disturb” a model. So, like a kid with a spinning gyroscope who cannot just sit there for long periods of time and watch it, but will rather disturb it with his finger. You too can influence your running model with a Force vector. This Force is a tool that can be activated in the Drawing Tool box. If you then move the cursor into the model window, you are in a position with the mouse to pull directly a preselected object of your model with this Force vector during the running simulation. The direction and the size of this Force vector is directly adjustable.

The keys “1”, “2”, “3”, ... of the keyboard have a “amplifier” function on the abolute size of your Force (see *User Manual*).

- “1”: 10 times
- “2”: 100 times
- “3”: 1000 times
- a.s.o.

64 Object Collision

1st rule:

During a simulation, the program can direct the entire collision-control check and can treat all contacts of objects automatically, without doing anything. So, you can generally leave the collision-control to the program.

2nd rule:

You can override this automated control anywhere it seems necessary with a great number of specific measures. You can appropriately set collision control in general, as well as determine specific properties for certain objects or for certain object pairs.

So, for example, you can set or influence the following properties.

Should object-pairs collide?

- Turn on/off collision control for individual object-groups.
- Turn on/off collision control for individual objects.
- Turn on/off bilateral collision control between two specific objects.
- Make collision control for individual objects active or passive (*).

(*) In collision control, only object-pairs in which at least one of the two objects is active will be checked.

How and with which physical properties should objects collide?

- Set general collision properties.
- Set collision properties for object-groups.
- Set collision properties for individual objects.

The properties mentioned here can be set or combined at the same time, in which case the priority of the various properties will determine which influence is correspondingly dominant for the individual objects. It is outside the scope of this introduction to discuss all the possibilities for collision control and to explain all the system-internal automation. Naturally, you can simply always use the automatic collision control and not worry about this topic any more. For simple models this is a passable way that works well. For complex models with several hundred objects, you can increase program performance considerably if an eye needs to be kept on collision control too. This is why a relatively demanding collisions control system has been integrated into the program. For further information, please see the User's Guide. Just two examples will be mentioned here.

Example 1:

Model type: All or almost all objects are moving randomly through the space. Each object can collide with any other object.

- Collision control is generally activated for all objects.
- The bilateral collision control that ”rules” over objects that either overlap or ”exist” in different levels is de-activated.

Example 2:

Model-type: Only one object moves freely. All other objects are either fixed or limited in movement to the extent that they could never touch each other. The free object can however collide with all objects.

- Collision control is generally switched on for all objects.
- The collision control is generally passive for all objects.
- The collision control is only active for the object that is able to move freely.

In a model like ”pin-ball” there are hundreds of objects, but only one object can get into collisions (the pin ball). Therefore, it makes no sense to continually check to see whether or not any one object has collided with any one other. It’s enough to see whether or not the ball has collided with another object.

65 Collision-Control Management

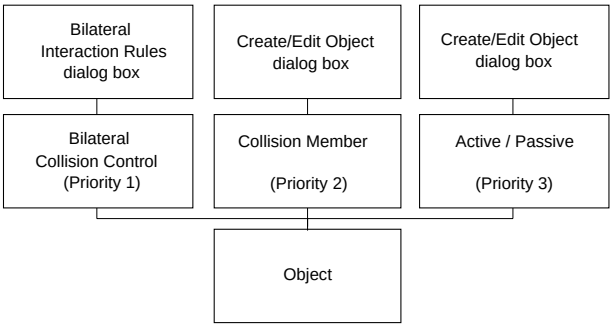


Abb. 57: The influence of different collision controls on an individual object.

The individual rules can be set and managed with the following dialog boxes:

A. Collision Control for individual objects.

- Create object dialog box
- Edit object dialog box
-

B. Bilateral Collision Control

- Bilateral Interaction Rules dialog box

Bilateral rules between two objects have the highest priority. If a bilateral rule has been set between two objects, the object-specific and the general rules will be ignored.

66 Perturbations

A running simulation can be disturbed in many ways while it's playing. Perturbations are always outside forces that affect individual objects, groups of objects, or all objects. Many different perturbations that overlap each other in regard to effect and distribution can be defined. Perturbations not only work to make the simulation more like physical reality, but are also quite useful to test and research the behavior of control systems under the influence of outside factors.

We differentiate between the temporal occurrence of the perturbation and the actual perturbation function.

Temporal Behavior:

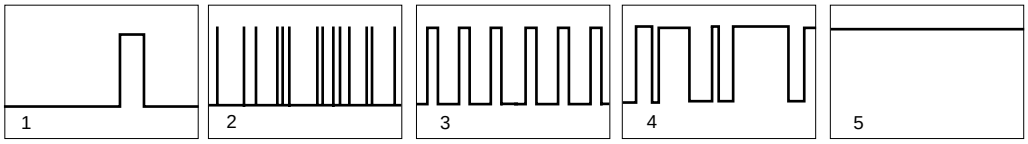


Abb. 58: Temporal behavior of perturbations

- 1. One-time perturbations
- 2. Discrete perturbations
- 3. Periodic perturbations
- 4. Random perturbations
- 5. Continous perturbations

Perturbation Function

You can create the perturbation function by combining given trigonometric and other functions. The actual perturbation signal is created in the end by the multiplication of the perturbation signal with the temporal occurrence.

Effective Direction:

Perturbations can behave in the following ways in regard to their spreading:

- in a specific spatial axis
- in all directions

silux provides a dialog box for the definition of continuous perturbations..

67 The Perturbation Dialog Box

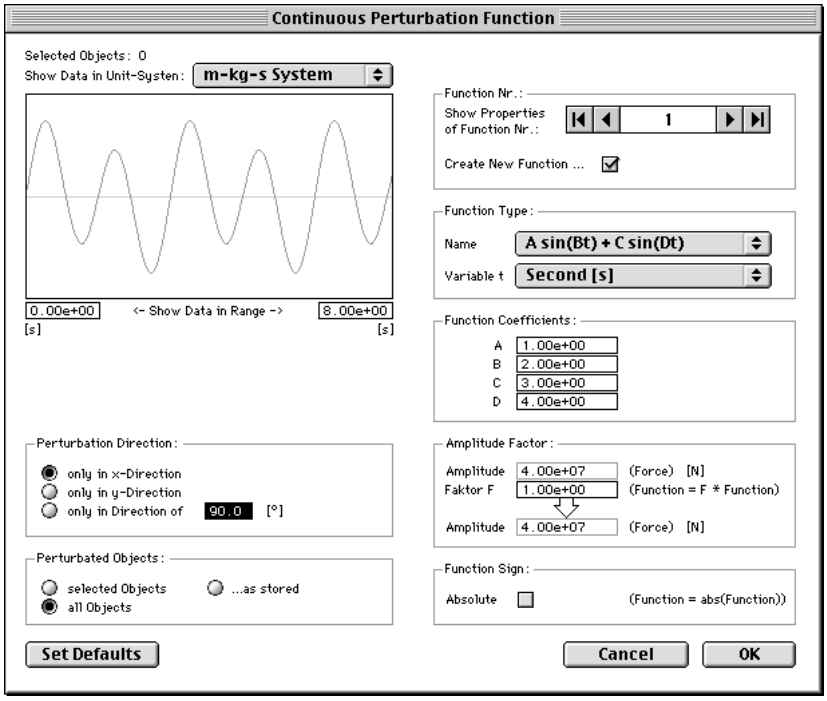


Abb. 59: Continuous Perturbation Function dialog box

The entries given in the right column characterize the curve-pattern, or the diagram, to the upper left. The two text fields directly under the graphics allow us to change the value of the representation in the dialog box in the horizontal direction. These two numbers affect the function itself but do not affect the results in your simulation: they just change the scale of the representation here in the dialog box.

In the perturbation direction field we specify in which direction the perturbation is effective. Only uni-directional perturbations are definable: a disturbance will always operate in one specific direction. To create a disturbance that operates in all directions, you need to define two perturbations; one in the y-direction and one in the x-direction. If the two perturbations are not identical they should have the desired effect, according to the principle of superposition.

Normal access to an existing perturbation is through the List Tool.

68 Friction / Overview

Mechanical friction (sliding friction, static friction) can be introduced in different grades in the silux program.

- Global friction properties (a single friction coefficient for all objects)

The program will calculate with the global friction coefficient.

The program always calculates sliding friction and static friction according to the classical formula:

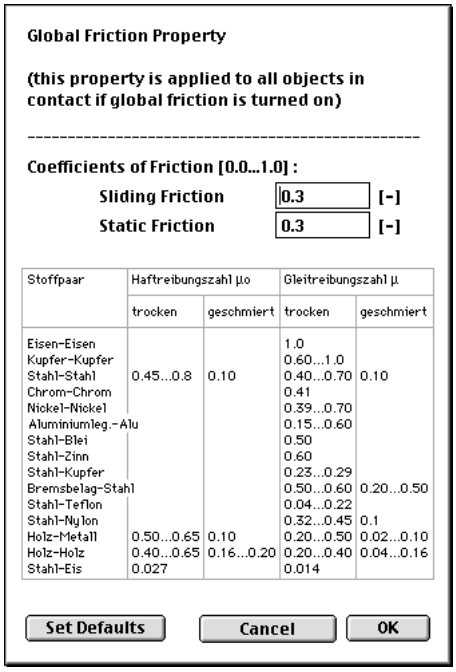
```
R = m N

R = friction
m = coefficient of friction
N = normal force between the two objects
```

Friction will be calculated and taken into account any time objects touch in the silux program. It is also taken into account upon impact.

The coefficients of friction for links are not affected by all of these settings. You can enter the friction torque for links in the Edit Link dialog box.

69 Global Friction Properties

Global friction properties with a single coefficient of friction are determined for all objects, no matter what sort of material the individual objects are made out of. The global frictional-coefficient can be switched on in the *Edit Global Friction* dialog box:



Stoffpaar	Haftreibungszahl µo		Gleitreibungszahl µ	
	trocken	geschmiert	trocken	geschmiert
Eisen-Eisen			1.0	
Kupfer-Kupfer			0.60...1.0	
Stahl-Stahl	0.45...0.8	0.10	0.40...0.70	0.10
Chrom-Chrom			0.41	
Nickel-Nickel			0.39...0.70	
Aluminiumleg.-Alu			0.15...0.60	
Stahl-Blei			0.50	
Stahl-Zinn			0.60	
Stahl-Kupfer			0.23...0.29	
Bremsbelag-Stahl			0.50...0.60	0.20...0.50
Stahl-Teflon			0.04...0.22	
Stahl-Nylon			0.32...0.45	0.1
Holz-Metall	0.50...0.65	0.10	0.20...0.50	0.02...0.10
Holz-Holz	0.40...0.65	0.16...0.20	0.20...0.40	0.04...0.16
Stahl-Eis	0.027		0.014	

Abb. 60: The Edit Global Friction dialog box

Global friction itself is turned on or off with the menu command *Enable Global Friction* or *Disable Global friction*. Friction can also be turned on and off or changed during a running simulation.

70 Control Systems

We are now moving to another level in regard to the use of the silux program. With what you have read so far, you are able to create a machine or a mechanical system and you are able to leave this machine to run on its own during the simulation. You can stop the simulation at any time, change the system and set it back into motion: you do not yet know however how to control the system during the actual simulation. This needs to change. There are basically two ways to control a system while the simulation is running:

- Interactive, manual control
- Autonomous, programmed control

1. Interactive, manual control

You can set off, change and control objects, propulsions and actuators manually in any way you like from outside. To do this, you can use the keyboard. If you want to use the keyboard for this purpose, you can assign certain tasks to specific keys. In the respective dialog boxes Edit Propulsion or Edit Actuator, other details, aside from these ”basic-wirings”, are available: how exactly the propulsion or actuator should react to a key signal, for how long, in which direction, etc.

With this interactive control, you really have the controls in your hands, in the true sense. This allows you to build simulators: simulators with which you can use your hands to directly control and influence the model on the screen.

2. Autonomous, programmed controls

The other way to control a model is the indirect way. We can define a control in the form of a program. All the programs that belong to a model are, at the time of execution, continually run and evaluated. There are essentially two ways to create a program:

- with the Create Control System dialog box

This is the simple method. In a purely interactive way, you can define a relatively simple control and assign the elements to be carried out, such as objects, actuators and propulsions.

- with a silux Script

You can type a program in the silux Script Language with the Script Editor or with an external word processing program.

71 Create Control System Dialog Box

The definition of a control system principally concerns the following:

- We want to select some physical variables from a running system.
- These variables should be transferred to a control-system so that a result can be calculated with a formula.
- The result should be used to control an actuator or a propulsion in that the resulting value would represent, for example, the operating value of the actuator or the directional effect of propulsion.

A control-system can be represented schematically as in the following:

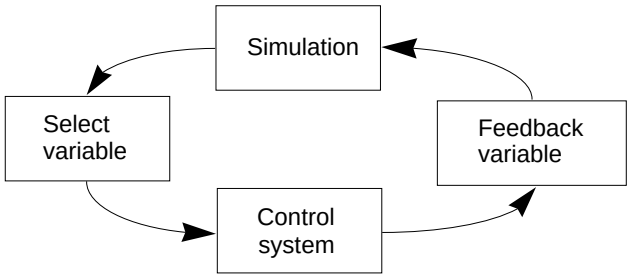


Abb. 61: The flow of data to the control-system and back

The entire control-system can be defined in the Create Control System dialog box. An already-existing control-system can be changed in the same dialog box. Access to the dialog box can go either through the menu (Create Control System) or through the List Tool Window if the control-system already exists.

It is our job to tell the system which variables we want to use as input for our control system in the Create Control System dialog box.: Which variables should then be further processed into a formula? The dialog box lets us enter up to three different variables for use in the formula to be defined. We speak of three so-called ”input channels” (X,Y,Z) that should not be confused with the area coordinates (x,y,z). Each input channel (X,Y,Z) can be assigned any physical variable. Similarly, the ”output” channel should be assigned a variable. This is the variable which resulting value is received from the formula, or from which variable we want to control the corresponding object. Finally, we define a formula of the type:

```
OUTPUT = Function (X,Y,Z)
```

In words: output variable = a function of the input variables.

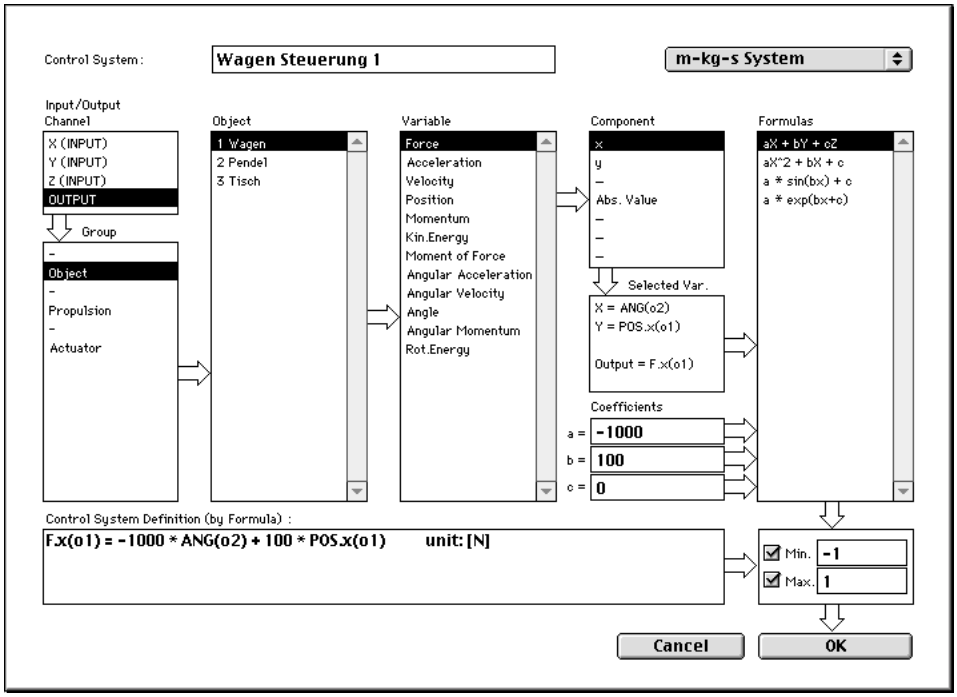


Abb. 62: The Create / Edit Control System dialog box

It is the task of the Control System dialog box to create the formula shown in the lower part of the dialog box in a comfortable way, only with pointing and clicking. The control-system that we're formulating is contained entirely in this formula. This formula contains all information which the control-system is made up of, as such. The formula defines a complete flow of data: where you should take a variable from, define what should be done with it and what should then be done with the final result.

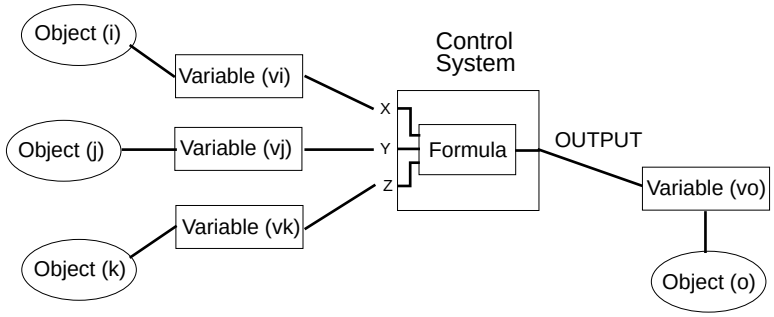


Abb. 63: The data-flow of a control system

If the capacity of the three input channels is not sufficient, we do not have to give up the project! There are two ways to bring more complex control systems into a model:

- You can assign a control-system to control many control systems. The supperposition principle holds here too.
- You can create the control system with a script.

Control System Scripts

Instead of building a formula in the Create Control System dialog box, we can just create a script that contains this formula as instructions. You can pack as many control systems as you want into one and the same script A control system in a script can also have more complex forms and it can, under certain circumstances, consist of multiple sorts of instructions. With a Control System Script we can tap the possibilities of silux Script. For more information about silux Script, please see the *silux script Language Guide*

72 Sensors

Sensors are probes or measurement receivers that have the ability to register a physical measurement and to forward it for connection with a gauge device. This forwarding can take place in electronic form or in another way. The gauge device can then process the reading.

There are different classes of sensors:

- Mechanical sensors
- Optical sensors
- Chemical sensors, etc.

In silux, we deal with mechanical sensors. Sensors come into the game when you want to know how large a physical variable is at a certain place. When we look at the physical variables of an object, then these values concern the object as a whole. If ask for the speed of an object, we will get the speed of a specific reference point on the object as the speed of the object as a whole. This point is the so-called reference point of the object. Generally though, each outer and each inner point of an object has a different velocity in relation to the global coordinate system. This is true for the amount as well as for the velocity vector. The speed of the reference point is in this sense a good average value of the object's velocity.

In order to determine a physical variable at a certain point of an object and to measure it over a certain period of time, you can fasten a small object (a sensor) onto the place in question. Typically, a small dot is fastened onto the point of measurement. If you want to measure the size and the duration of the force of impact during a collision of an object, then fasten the small dot to the point where it will feel the entire force of impact of the collision, that is so that the entire force passes over the sensor.

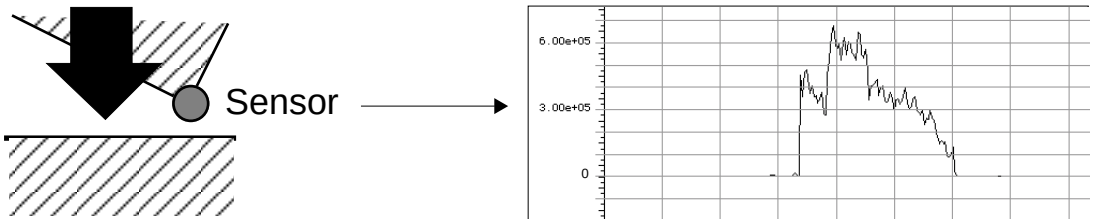


Abb. 64: Schematic representation of a sensor at a point of collision and display the force of impact during a collision.

After this, you don't have to do anything but watch the force on this sensor (object) in a Graphic Window that you've opened especially for this purpose.

Sensors work in silux just like they do in reality. If you really wanted to measure the duration

of force during a collision then you would install a force receiver at the measuring point and let the impact run over this sensor. We do exactly the same here.

You can use as many sensors as you'd like in a model. Sensors can also be added as needed during a pause in simulation, moved or simply removed again. Sensors can be made so small that they do not influence the or the mass proportions or geometry of the model. Sensors can also be attached to the inside of a model in order to, for example, follow the path of a certain point of an object or its velocity course.

The following physical variables are especially suitable for measurement with sensors:

- Force
- Acceleration
- Velocity
- Position

The measurements can be made in components in an area coordinate system or in absolute sums.

The measured values from sensors can also be used as input for control systems. So, you can also use local object conditions, not only object properties as a whole, in control-systems.

Aside from these sensors, you can also observe the physical properties of a link during a simulation. Links are already localizable, dot-shaped figures that are suitable for the questioning of force, acceleration, velocity and position. The forces in links are very often of special interest. Extra sensors do not have to be installed for the observation of these physical variables.

73 Micro-Movements

When you record short-lasting, intense events in physical reality, for example, a collision between rigid bodies, you won't generally record a nice smooth curve as you would from an idealized and simplifying theory. Generally, a great number of small and extremely small oscillations are superimposed over the idealized curve and this diffuses the signal along the ordinates. When you look at it, the signal looks as if random noise was superimposed over it (see chapter Sensors). In reality, in the parts of a real machine, an intensive impact would initiate many oscillations: these oscillations are scattered from many different surface areas in many different ways. Sometimes oscillations are reflected back to where they started from where they then cover the initial signal, and so on.

The signal of impact measured by a sensor in the silux program generally looks exactly this way. This is not because silux pays any attention to the inner process of oscillations (this is only the case in objects with internal grid structures) but rather because the part in question is generally connected to an entire chain of links with other objects. These objects will themselves oscillate, possibly delayed, and so, in the form of feedback, influence the original signal.

The silux program behaves the same as reality in regard to this problem. In reality, as in the silux program, the surface at the point of collision generally makes micro-movements or vibrations. It is important to realize that these vibrations are not the result of an inadequate measuring system, but rather that true measurement signal really looks like that in the case of large enough resolution. Physical reality is not smooth. You don't however make any great mistakes in interpreting the measurement signal if you do smooth the signal. In the case of overloading, it is not the micro-movements that are generally responsible for the demand on a machine part and its destruction as a whole, but rather the average course of the signal. Mirco-movements are strongly place-dependent and can be locally differentiated. They can appear quite different even in directly neighboring areas. Micro-movements are not then generally forces that affect machine parts as a whole, but rather local oscillations in the material structure.

You can use the following methods for the interpretation of such measurement signals:

- You can accept the signal as it is and read the average course of the curve from it intuitively.
- You can dampen the signal by reducing the size of the so-called "interaction constants" of the involved objects and sensors.
- You can dampen the signal afterwards, graphically, during plotting on the screen.

74 Measurement Resolution

To answer the question "What resolution can a silux simulation or, better put, a single event in a simulation, have?", we'll look at the example of a collision of two objects.

If we mark a single point on one of the two objects and write down its time-space conditions during the simulation, it will be clear that we can only note the position of this point at certain times (t). If we do this at the time (t), then the next possible chance to do it is at time (t+dt). The geometric position of the marked point at the points of time in question is though known exactly.

Because of this, we can calculate the position between the two points of time (t) and (t+dt) through interpolation. This does not however improve the resolution of the measurement within the time (dt). This is because if something happens in this time, the program notices it first only at time (t+dt). The program calculates with the time step (dt) and, after this period of time has passed, stores how and where the conditions have changed so that it can calculate the new values of all physical variables for the point of time (t+dt). Even if you tried to use interpolation to calculate back to the values at a more exact time period within (dt), the fine structure of the events within the time span (dt) would remain hidden to you.

The only way to improve the resolution of a single event would be to decrease the time step (dt) during the event. The resolution of the measurements should not however be a problem in silux because the time steps for laboratory scales is on an order between a millisecond and a microsecond. The resolution is generally much better than you'd expect and better than what you need for practical uses. More problems come from the resolution of a signal having too many small-scale structures so that you have to smooth the signal out before being able to work with it any more.

In general, the time step in the silux program is automatically determined by the smallest geometric and physical structures available. In this way we can make sure that the course of movement of these small-scale structures is developed in all its details over the course of time.

75 Coordinate Systems

Cartesian coordinate systems are primarily used in the silux program. We differentiate between two types:

- Global coordinate system
- Local coordinate system

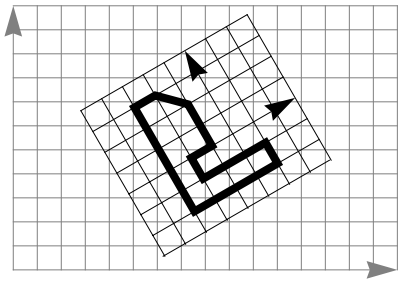


Abb. 65: Object with local coordinate system, imbedded in a global coordinate system

The global coordinate system

The global coordinate system covers the entire simulation area. The coordinates of the global coordinate system are shown along the upper and left-hand sides of the model window. The global coordinate system is used to indicate and save the positions of elements, objects, etc. When talking about the position of an object, you're always referring to coordinates of the point of reference of the object.

The local coordinate system

A local coordinate system is an object-specific coordinate system which is closely tied to the corresponding object and moves and rotates with it. Each object has its own local coordinate system. Only the reference point of the object is saved in the global coordinate system. All other object-specific coordinates are relative to this point of reference, to be understood in relation to the object-specific coordinate system. These local coordinates do not then change when the object moves and rotates in space.

For further information about the coordinate system, please see: *User Manual*

76 Unit Systems

We differentiate fundamentally between the unit system that silux calculates with and different unit systems with which the program represents itself and with which data can be entered into the dialog boxes.

- Calculation system (lab scale: cm-g-ms system)
- m-kg-s system
- cm-g-ms system
- cm-g-s system

The calculation system is a concern of the program itself. The program will decide independently which system of calculations is most appropriate for a specific model.

Everywhere you have to enter unit-dependent numbers, you can generally decide yourself what units you want to use. Generally, there's a pop-up window in such dialog boxes where you can chose which unit system you wish to use. You can activate the appropriate system for each individual number-entry. Each time you change the unit-system in a dialog box, all numerical values in the dialog box will be calculated into the new values and will then, correspondingly, be represented to show the new values.

77 Saving Models

A silux model can be saved on an external disk in a number of ways. The following file formats are available:

- The SILUX file format
- TEXT file and exchange format
- DXF-exchange format
- MACRO file format (silux Script)

SILUX File format

This is the standard file format for saving models for every individual user and for every user operating within a local area network. If you chose the command ”Save” in the silux program a silux file will be created if nothing else has been explicitly chosen.

TEXT File and exchange format

The TEXT format is useful for transporting models over different platforms with different operating systems. The TEXT format is not sensitive to version changes of the programs and can, without being changed and without any conversion, be loaded by future or old program versions.

DXF-exchange format

This is the most commonly used exchange format worldwide. Practically every CAD program can load and save this format. This allows you to join silux with already-installed basis of CAD systems. This file format is generally used to transfer files from other drawing or CAD programs.

MACRO format (silux Script)

Instead of saving a model itself, you could also save the instructions for constructing the model. If a model is going to be tested in many versions and variations, it may be easier and quicker to carry these modifications out in a macro.

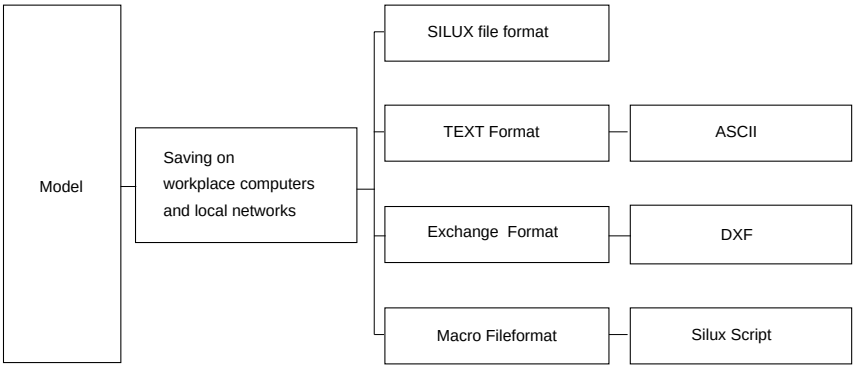


Abb. 66: Model-saving possibilities with silux

78 Notes